

10. Off-Line Services

10.1 Real Time Analysis

10.1.1 Clock Correlation

The Clock Correlation process is responsible for evaluating the bias between the time kept by the onboard spacecraft clock and the UTC time kept on the ground system. This bias, or clock error, is evaluated by two separate and independent algorithms: the Return Data Delay (RDD) and the User Spacecraft Clock Calibration System (USCCS) techniques.

The process connects to a real-time string upon startup. The process is started with an ECL directive with the following syntax:

```
CLOCK CORRELATION STARTUP STRINGID=#  
SCID=#MODE=#
```

where:

STRINGID is an integer value describing the real-time string to which the process must connect;

SCID is the spacecraft ID (e.g., AM1)

MODE is OPS, TRAINING or TEST

- a. **Clock Controlling.** Rather than using clock jumps, the spacecraft clock will be steered by adjustments to the Master Oscillator frequency. This change in frequency adjustment will occur when the clock bias has exceeded some predetermined positive value.
- b. **Return Data Delay (RDD).** The RDD method calculates the clock bias by using the time stamps on the telemetry packets. Various time delays are subtracted from the Ground Receipt Time to determine an estimated telemetry packet time. This estimated packet time is compared to the actual Spacecraft Time, the difference between the two being the RDD clock bias.
- c. **Implementation.** Due to the inherent inaccuracies of various time stamps and time delay measurements in the EOS configuration, the RDD method is guaranteed to generate clock bias measurements accurate to only 20 milliseconds. During the initial phase of a spacecraft mission (e.g., during orbit acquisition), this will be a sufficient accuracy specification to provide information about the spacecraft clock. It will allow the FOT to steer the clock to 20 millisecond accuracy, at which point the USCCS algorithm will be able to begin to function.

In nominal operations during a real-time pass, the RDD method will announce calculated a clock bias for each passing minute, via an event message. Once the contact is over, the process will determine an average RDD calculated bias for the entire pass. This average value is reported to the FOT through an ASCII report and an event message. If a spacecraft's mission requirements for onboard clock accuracy are finer than the 20 millisecond inaccuracy of the RDD method during nominal operations (e.g., AM1's 100 microsecond accuracy is finer than the 20 millisecond inaccuracy of the EOS

configuration), the clock bias will be truncated by the inherent inaccuracy, and thus will contain a zero-value. These minute-by-minute calculations are to be used as "sanity checks" for nominal operations.

- d. **Return Channel Time Delay (RCTD).** The RDD method uses as input an RCTD message from NCC, which comes to the EOC as an OPM-62. This message contains time delay measurements describing the amount of time required for the telemetry signal to travel from the TDRS ground station to EDOS. This message class is received at the end of a real-time contact, and its contents are stored as input for the RDD method running for the subsequent real-time contact.
- e. **User Spacecraft Clock Calibration System (USCCS).** The USCCS method makes use of Pseudo-Noise (PN) Ranging Epochs generated by a TDRS Tracking Service. These epochs are transmitted from the TDRS Ground Terminal (TGT) at a rate of 11.75 per second, one per 85 milliseconds. Once a forward epoch is received by the spacecraft, the transponder sends a Time Transfer Epoch to the CTIU, which in turn places a spacecraft clock reading into the telemetry stream. Once the Time Transfer Epoch is generated, the transponder sends a return PN epoch to the TGT.
- f. **Time Transfer Message.** The forward and return ranging epochs are time-stamped and recorded at a rate of one per second. The recorded epochs are sent to the NCC, which in turn forwards them to the EOC as a Time Transfer Message (TTM), also known as OPM-66. This message also contains the forward and return Range Zero Set (RZS) time delays. These delays describe the amount of time the forward and return epoch signals spent within TGT and TDRSS hardware for the given tracking service.

The Clock Correlation process receives the TTM at the end of the tracking service. Once the process receives the recorded epochs, it reconstructs the unrecorded epochs. The process then begins matching forward and return epochs to the telemetry packet containing the spacecraft clock reading associated with that epoch pair. A USCCS clock bias will be calculated for each telemetry packet containing a spacecraft clock reading. The process generates an average of all the USCCS calculations, and reports them to the EOC via an ASCII report and event message.

- g. **Automated Anomaly Processing.** In the event the USCCS-calculated bias exceeds some predetermined threshold, the Clock Correlation process will determine that a frequency adjustment is required. The process by default will generate a Command Request to update the clock, as well as a TONS table load to be uploaded to the clock. The Command Request is submitted to do the actual adjustment, while the table load is generated to notify TONS of the impending adjustment. These products are then posted for approval/disapproval by the Command Activity Controller.

This automated functionality can be enabled or disabled by an ECL directive, using the following syntax:

```
CLOCK CORRELATION COMMAND REQUEST ENABLE
.....DISABLE
CLOCK CORRELATION TABLE LOAD ENABLE
.....DISABLE
```

10.1.2 Solid State Recorder Manager

The Solid State Recorder (SSR) Manager will supply Command Requests to the user to efficiently play back and replay data from the Solid State Recorder buffers. The SSR Manager will display the status of the SSR buffers during the contact. This status will include problem and solution messages, as well as missed data locations. Refer to Section 9 for additional information on employing real-time services to monitor and control historical replay activities for a given spacecraft.

To start the Solid State Recorder Manager:

Figures 10.1.2-1 through 10.1.2-4 present displays associated with the SSR Manager.

1. To start the SSR Manager, type the following command from the allocated SSR host:
SSRStartUp
2. From the Control window command line enter the following ECL directive:
EA ENABLE - stringId <real time string>
3. Click **Tools...**
4. Select **Schematic Display**.

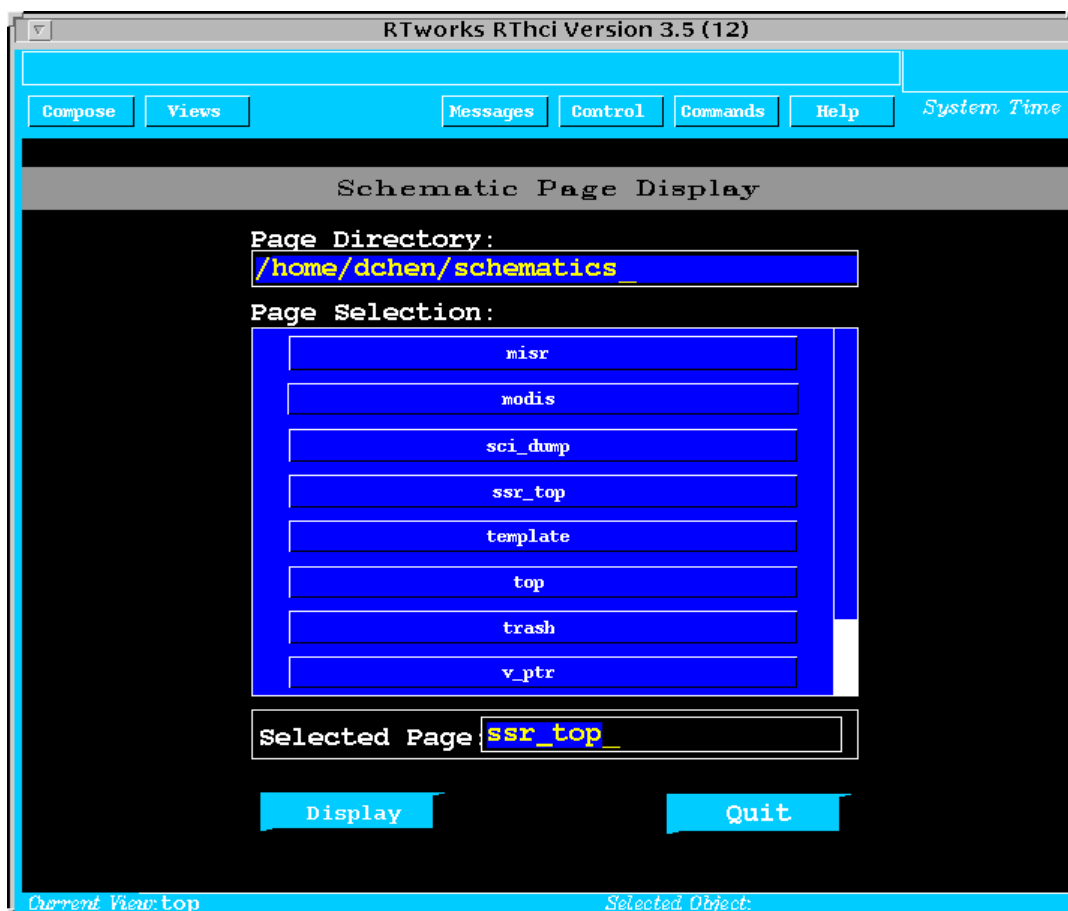


Figure 10.1.2-1 Schematic Display Window

RTWorks opens (see Figure 10.1.2-1).

1. Click **ssr_top** or type `ssr_top` in the Selected Page text entry box.
2. Click **Display** or press <Enter>.

The SSR Manager schematic page opens (see Figure 10.1.2-2).

Click on the individual buffer row to bring up the individual buffer window.

For example, if the MODIS Buffer is selected, the MODIS Buffer Schematic Page opens (see Figure 10.1.2-3).

The **BACK** button brings you back to the multiple buffer window.

The **H&S Data** button brings up the AM1 SSR Health and Safety Data window (see Figure 10.1.2-4).

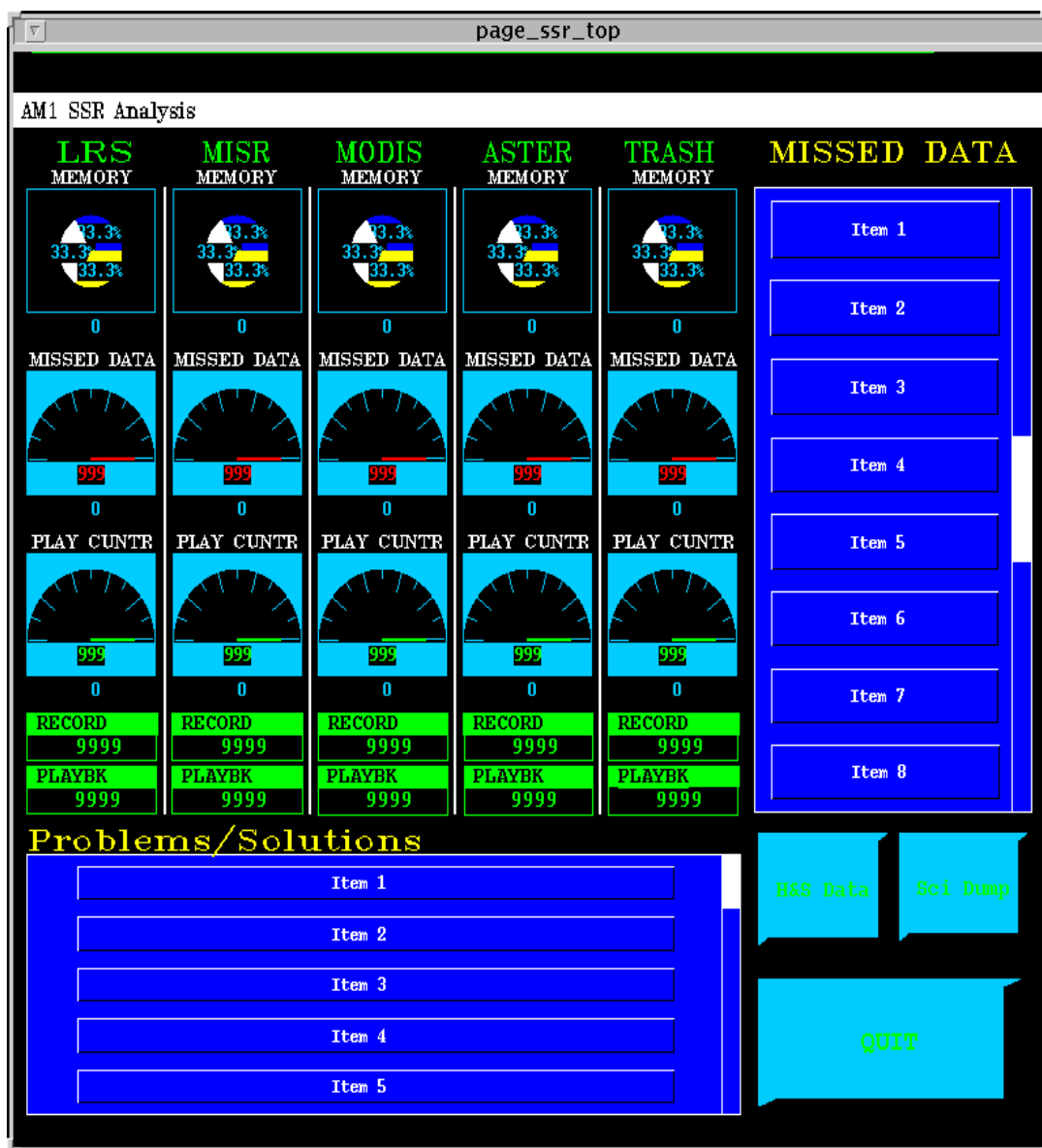


Figure 10.1.2-2. SRR Manager Display Window

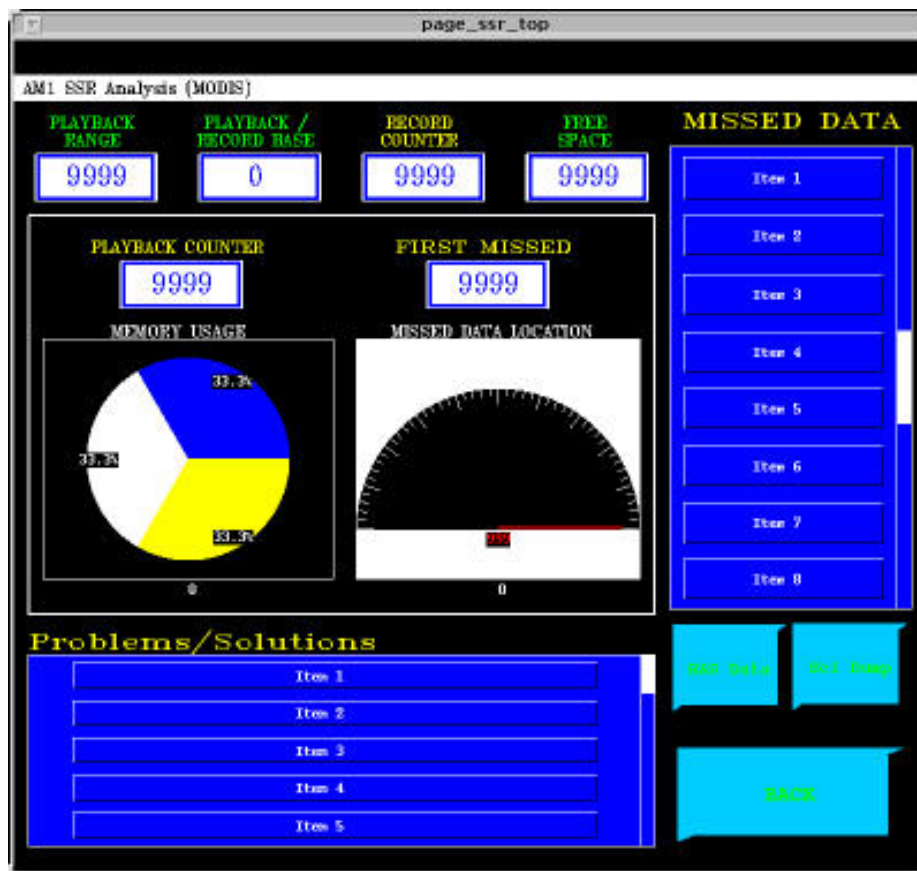


Figure 10.1.2-3. MODIS Buffer Display Window

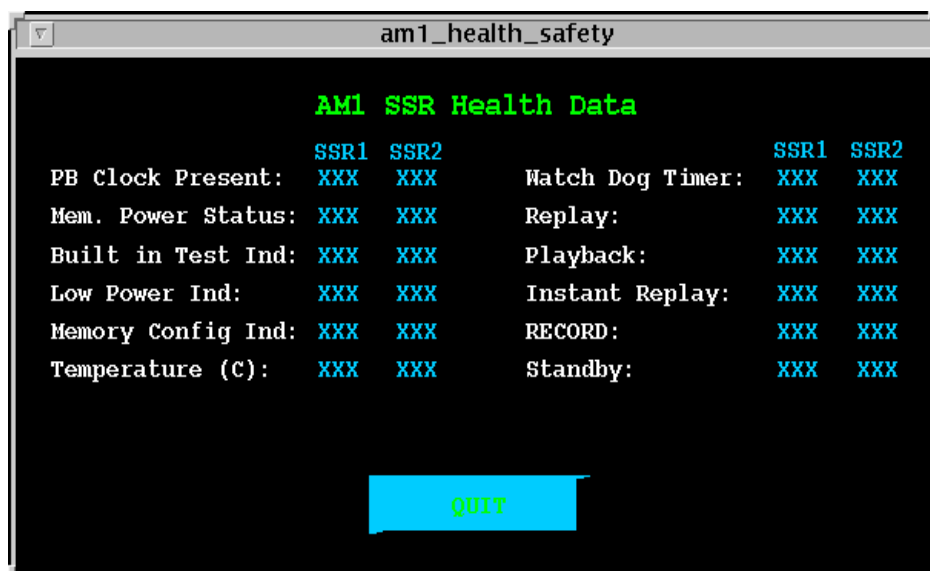


Figure 10.1.2-4. AM-1 SRR Health and Safety Data Window

- a. **Configuration Files.** The configuration files described in the following text are used by the SSR Manager to build the playback and replay command requests. The objective is to replay and playback as much data as possible during the contact.

1. **The percentage.cfg file:**

The **percentage.cfg** file is used if the total amount of data recorded for all the buffers cannot be played back. Percentages for each buffer can be set. The SSR Manager will then try to playback a percentage of each buffer using these values.

The file is an ASCII file containing five numbers delimited by a space. The order of the buffer percentages is: LRS, Misr, Modis, Aster, and Trash.

For example:

.95 .75 .75 .60 1.0

95% of LRS will be played back.

75% of Misr and Modis.

60% of Aster.

2. **The bufferOrder.cfg file:**

The **bufferOrder.cfg** file is used if all the data cannot be played back during the upcoming pass or the percentages of the buffers cannot be played back during the upcoming pass. The file sets the order of the buffers to be played back.

This file is an ASCII file containing five numbers delimited by a space. These numbers are the buffers Buffer ID.

The Buffer ID's are: Trash0, LRS1, Misr2, Modi3, and Aster4

For Example:

3 2 1 4 0

The new order will be Modis, Misr, LRS, Aster, and Trash.

3. **The thresholds.cfg file:**

The **thresholds.cfg** file sets the replay percentage thresholds for each buffer. The percentage represents the limit or amount of data to be replayed compared with the size of the buffer that has been played back. Once this threshold has been reached, one replay will be issued for the amount of data that was previously played back, instead of multiple replays for each area missed. The file is an ASCII file containing five numbers. The order of the buffer threshold percentages is: LRS, Misr, Modis, Aster, and Trash.

For Example:

..40 .65 .65 .75 .30

If 40% of the LRS buffer needs to be replayed, replay all of the played back buffer.

65% of Misr and Modis.

75% of Aster.

30% of the Trash buffer.

10.1.3 The Decision Support System (DSS)

The Decision Support System (DSS) will monitor the health and safety of AM-1 through telemetry and ground support data. The DSS will determine the current state of the spacecraft and track spacecraft state changes. The system will detect and isolation anomalies, and provide recovery recommendation to the user. It will also generate event messages and logs as part of its output to the user.

To start the DSS:

1. Telnet to "astro."
2. Set display back to current userstation or xterm.
Source the Altair setup file: `/net/astro/data/mcs_2.01/config/scripts/app_env_setup`
3. Change to Altair home directory: `cd $MCSHOME`
4. Increase available system resources for Altair processes: `unlimit descriptor`.
5. Change to inference engine directory: `cd apps/ie`
6. Start RTworks inference engine process in the background: `rtie &`
The RTie interface comes up and will already be in run mode (it takes a couple of minutes to completely initialize).
7. Start Altair manager process in the background: `mcs &`
A MCS Management window will pop up with a set of radio buttons to select the Altair processes to start.
8. Click **RTworks Server**, **IGE**, and **IE** buttons, so as not to start those processes as they are not currently needed.
9. Click **Start MCS** to start the necessary Altair processes (it take a couple of minutes). Three windows will pop up; Message Handler (MH) window which outputs state changes and other state information; State Transition Engine (STE) window which can be used to view STE status, and execute procedures; and the Procedure Automation Language Shell (PALS) window which is used to view SRE and other process status, states, and execute procedures.

NOTE

Steps 1 and 3 will change if Altair or RTworks resides on a userstation other than "astro."

a. DSS Windows

1. **Altair MCS Management Window.** The Management window allows you to: activate and terminate Altair processes; control where the processes execute; control where the Altair displays are sent; and set the level of debug messages generated in the log files (see Figure 10.1.3-1).
2. **Message Handler Window.** The message handler window is Altair's version of the FOS event handler for all Altair-generated messages. It will have the window title "MH." The message format displayed in the window is divided into three main parts.

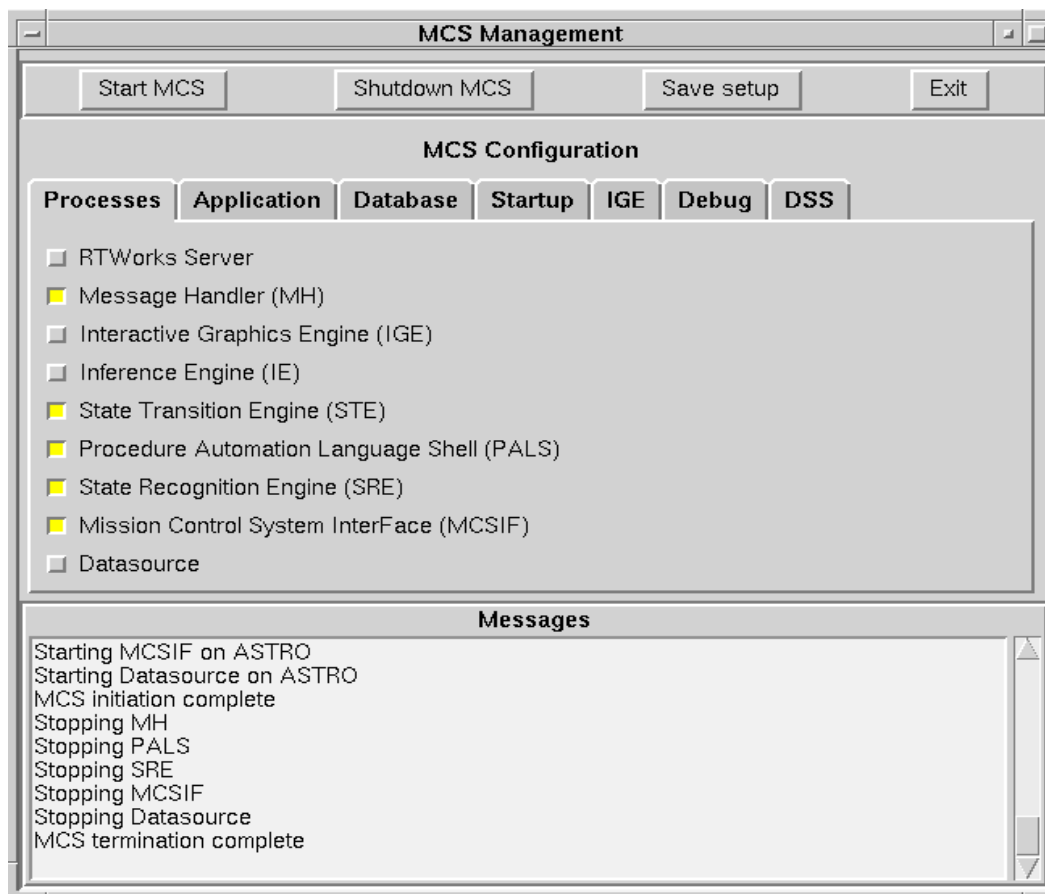


Figure 10.1.3-1. MCSM Window

The first part displays the initial of the process name that generated the message (i.e., SRE for State Recognition Engine, STE for State Transition, PALS for Procedure Automation Language Shell,...). The second part is the time stamp of the message. The third part is the actual text of the message. Messages displayed include state changes, procedure execution and status, and PALS command execution and status (see Figure 10.1.3-2).

3. **DSS Control Window.** This window provides the capability to display the current states of all systems defined in Altair; the current value of the telemetry data used by the systems; view the SRE log; and print the SRE log (see Figure 10.1.3-3).
4. **State Transition Engine (STE) Window.** This window provides a command line prompt to execute tcl/tk based procedures. The window title is "AMCS STATE TRANSITION ENGINE Main."
5. **Procedure Automation Language Shell (PALS) Window.** The procedure automation language shell window provides a command line prompt to execute PALS commands. The window title is "PALSCON" (see Figure 10.1.3-4).
6. **RTworks Inference Engine (RTie) Window.** This window displays state transition related to rules firing (see Figure 10.1.3-5).

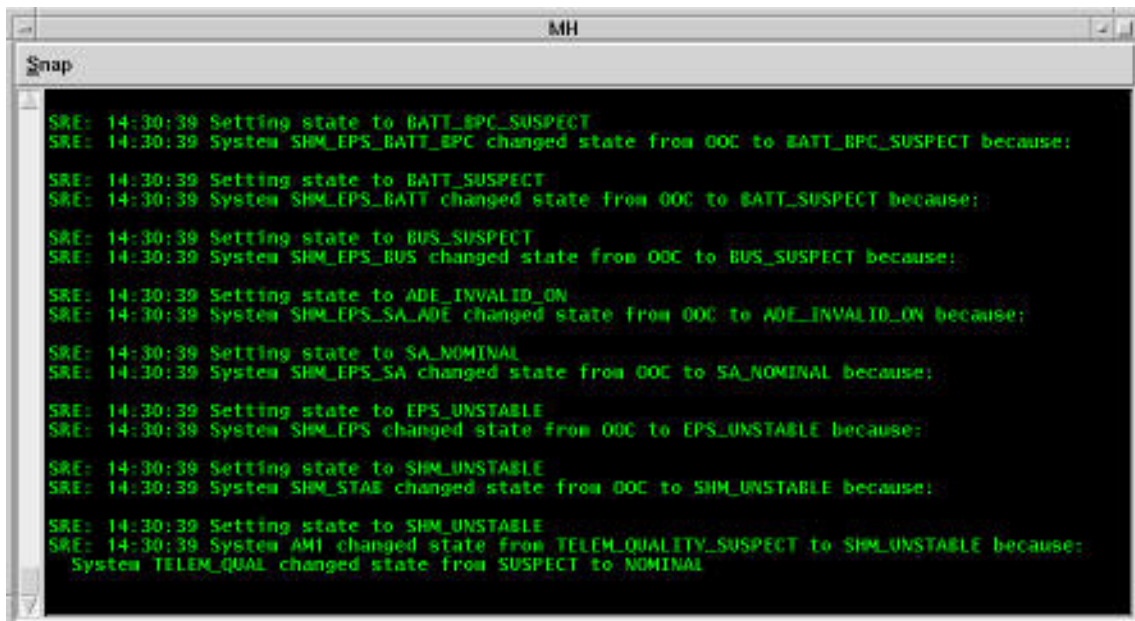


Figure 10.1.3-2. MH Window

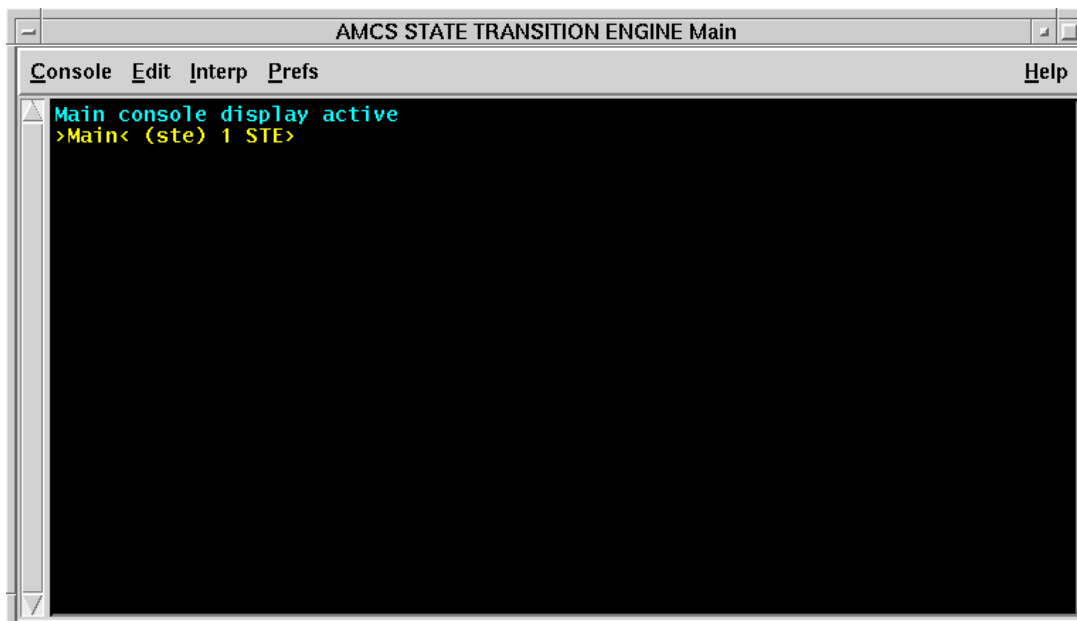
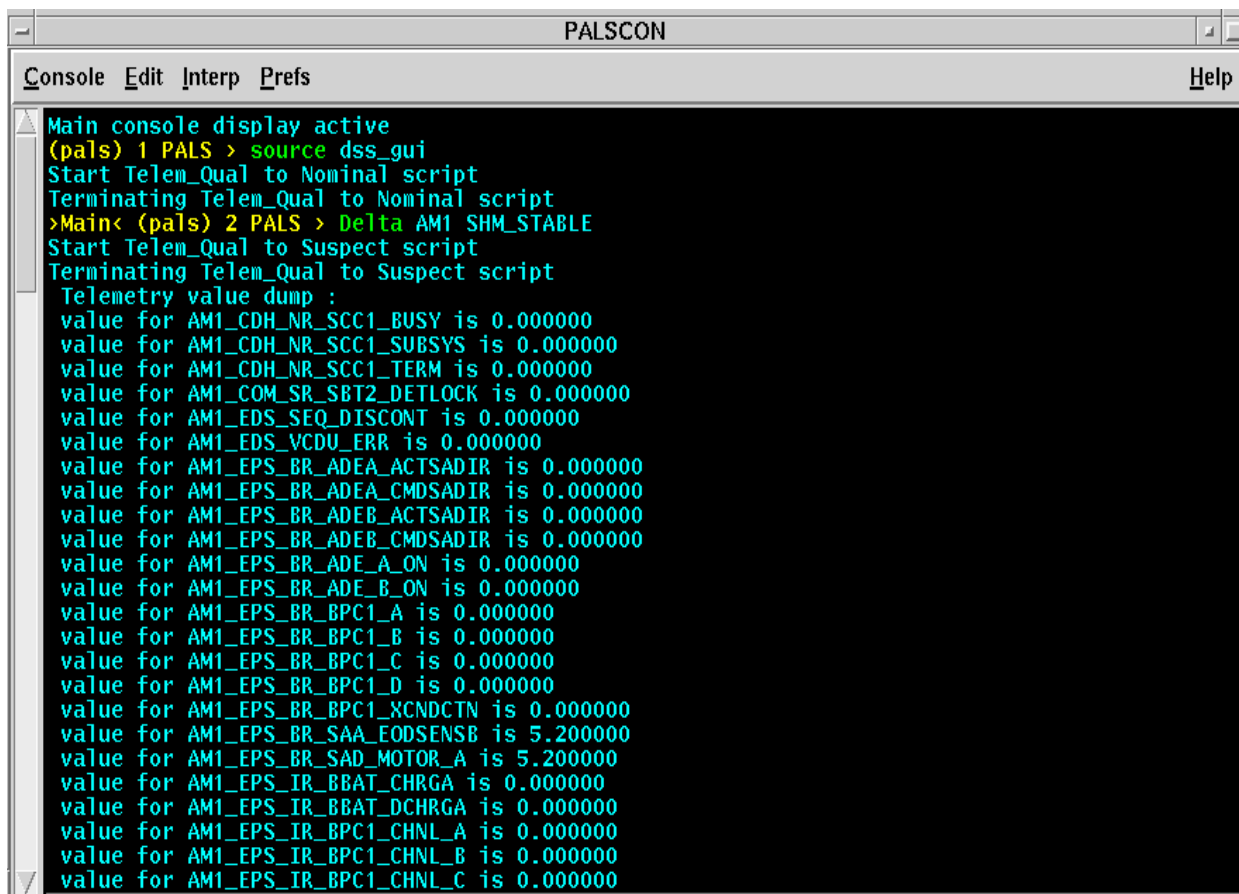


Figure 10.1.3-3. AMCS STE Window



```
PALSCON
Console Edit Interp Prefs Help
Main console display active
(pals) 1 PALS > source dss_gui
Start Telem_Qual to Nominal script
Terminating Telem_Qual to Nominal script
>Main< (pals) 2 PALS > Delta AM1 SHM_STABLE
Start Telem_Qual to Suspect script
Terminating Telem_Qual to Suspect script
Telemetry value dump :
value for AM1_CDH_NR_SCC1_BUSY is 0.000000
value for AM1_CDH_NR_SCC1_SUBSYS is 0.000000
value for AM1_CDH_NR_SCC1_TERM is 0.000000
value for AM1_COM_SR_SBT2_DETLOCK is 0.000000
value for AM1_EDS_SEQ_DISCONT is 0.000000
value for AM1_EDS_VCDU_ERR is 0.000000
value for AM1_EPS_BR_ADEA_ACTSADIR is 0.000000
value for AM1_EPS_BR_ADEA_CMDSADIR is 0.000000
value for AM1_EPS_BR_ADEB_ACTSADIR is 0.000000
value for AM1_EPS_BR_ADEB_CMDSADIR is 0.000000
value for AM1_EPS_BR_ADE_A_ON is 0.000000
value for AM1_EPS_BR_ADE_B_ON is 0.000000
value for AM1_EPS_BR_BPC1_A is 0.000000
value for AM1_EPS_BR_BPC1_B is 0.000000
value for AM1_EPS_BR_BPC1_C is 0.000000
value for AM1_EPS_BR_BPC1_D is 0.000000
value for AM1_EPS_BR_BPC1_XCNDCTN is 0.000000
value for AM1_EPS_BR_SAA_EODSENSEB is 5.200000
value for AM1_EPS_BR_SAD_MOTOR_A is 5.200000
value for AM1_EPS_IR_BBAT_CHRGA is 0.000000
value for AM1_EPS_IR_BBAT_DCHRG is 0.000000
value for AM1_EPS_IR_BPC1_CHNL_A is 0.000000
value for AM1_EPS_IR_BPC1_CHNL_B is 0.000000
value for AM1_EPS_IR_BPC1_CHNL_C is 0.000000
```

Figure 10.1.3-4. PALSCON Window

- b. **DSS Logs.** State Recognition Engine (SRE) Log - The ASCII log file contains event messages generated by Altair's State Recognition Engine process. All state transitions, and PALS command/procedure status will be included in the log.
- c. **DSS Execution.** As the values of telemetry points monitored by the state model changes, state changes are logged in the message handler. The value of the telemetry points can be changed either externally (through the parameter server) or internally (from the PALSCON window) through PALS commands or procedures.

10.2 Off-Line Analysis

The basic Analysis Request allows you to create a dataset of telemetry covering time spans of valid archived housekeeping data. The basic request allows you to sample telemetry by every N samples, where N is a number from 1 to 32767, or by change only. You can plot data contained with a dataset and create reports containing plots and tables.

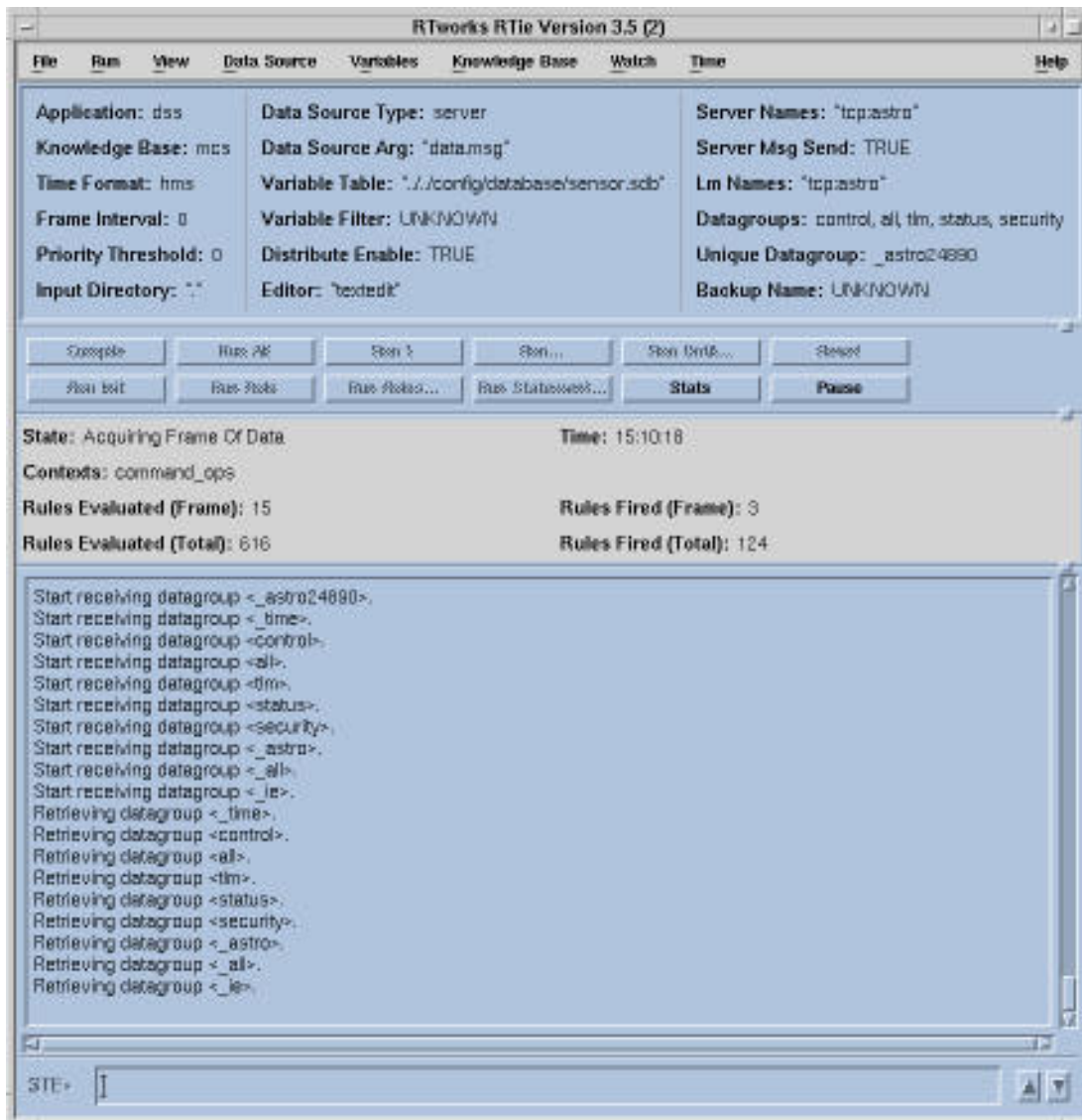


Figure 10.1.3-5. RTie Window

User Selectable Statistics are minimum-maximum-mean standard deviation statistics calculated over an interval between 1 second and 24 hours. You may select interval lengths for user statistics while building the basic analysis request. The resulting data is placed in a dataset that may be formatted as a graph or table.

10.2.1 Build an Analysis Request

To build on analysis request:

1. Open the Analysis Request Builder Window (see Figure 10.2.1-1).
2. Enter a request name in the text field next to Request Name.
3. To Select Start and Stop Times for the data to be analyzed:

Analysis Request

File Help

Request Name: Request Status: Submitted

Request Processing Site

- ☒ Local Only
- ☒ EOC Only

Data Quality

- ☒ Good Data Only
- ☒ All Data

Selected Times

Start Time	Stop Time
1997/140 19:08:01	1997/140 19:09:01

Selected Telemetry

Parameters	Sampling Rate	Statistics Rate
SDU_SCTIME	All Data	N/A
EDS_GRT	All Data	N/A

Select Time... Standing Order... Select Telemetry...

Product Options

- ☐ Telemetry Attributes
- ☐ Graph
- ☐ Table
- ☐ Output DataSet Name
- ☐ Input DataSet Name
- ☐ CarryOut File Name

OK Algorithm... Cancel Help

Figure 10.2.1-1. Analysis Request Builder Window

- Click Select Time to open the Pair Time Selector window (see Figure 10.2.1-2).
- Modify if necessary, the start date (i.e., 1996/231); press <Return> to accept this new start date.
- Modify if necessary, the start time (i.e., 17:44:32.000); press <Return> to accept this new start time.
- Modify if necessary, the end date (i.e., 1996/232); press <Return> to accept this new end date.
- Modify if necessary, the end time (i.e., 17:44:32.000); press <Return> to accept this new end time.

- f. Click OK to accept this pair time to be used in the analysis request. The pair time is now selected and displayed in the table labeled Selected Times on the Analysis Request Builder window.
- g. Select another pair time if necessary, by repeating steps 1 through 6.

NOTE

The **Absolute** and **Relative** selection boxes will invoke different dialog boxes. The **Time** and **Event** selection boxes may be user-defined as relative or absolute. Examples of absolute and relative time and events include: absolute time - February 14, 1997; relative time - the first day of every month; absolute event - sunrise on February 12, 1997; relative event - sunset every first day of the month.

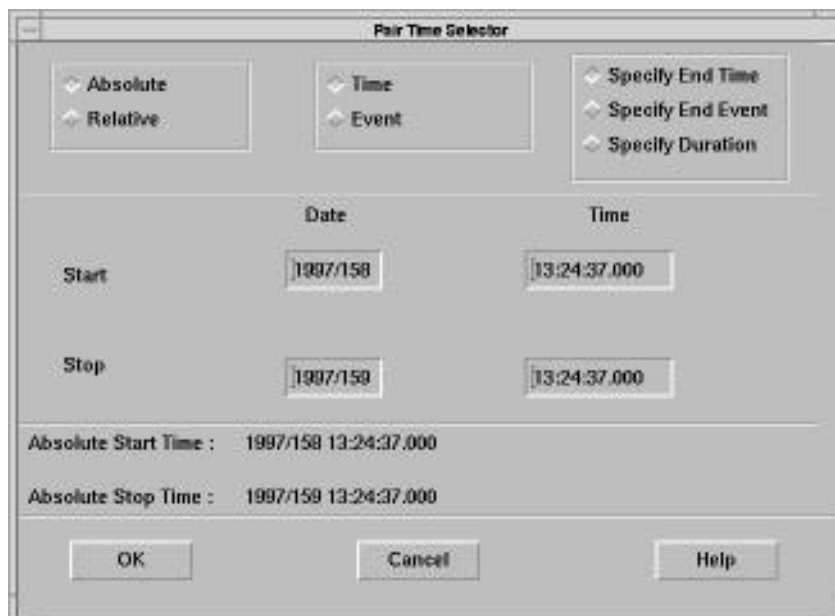


Figure 10.2.1-2. Pair Time Selector Window

4. **To select a parameter with sampling and statistics rates:**
 - a. Click Select Telemetry to open the Analysis Telemetry Selector window (see Figure 10.2.1-3).
 - b. Select a subsystem path.
 1. Click Filter to open the Selection Filter window (see Figure 10.2.1-4).
 2. Click a spacecraft subsystem (i.e., AM1) in the Spacecraft list.
 3. Click on an Instrument/Subsystem (i.e., GNC) in the Instrument list.
 4. Click B in the third list box. Click on a sample type (i.e., B)
 5. Click Select to select this subsystem mnemonic. The subsystem mnemonic ("AM1_GNC_I") is now selected and displayed in the right most list box labeled Selected.
 6. Select another subsystem mnemonic, if necessary, by following the instructions indicated in steps 1 through 6.

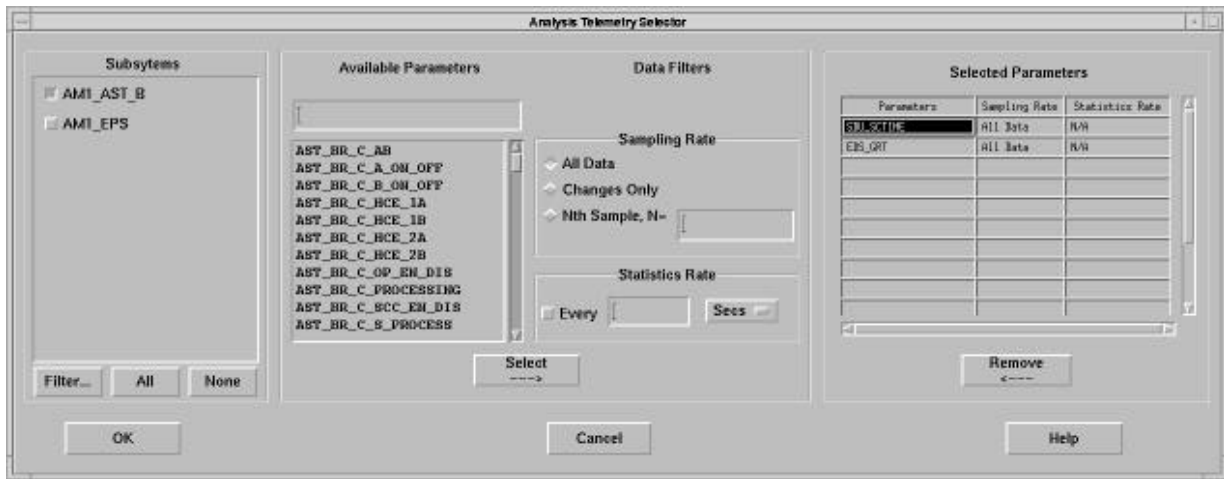


Figure 10.2.1-3. Analysis Telemetry Selector Window

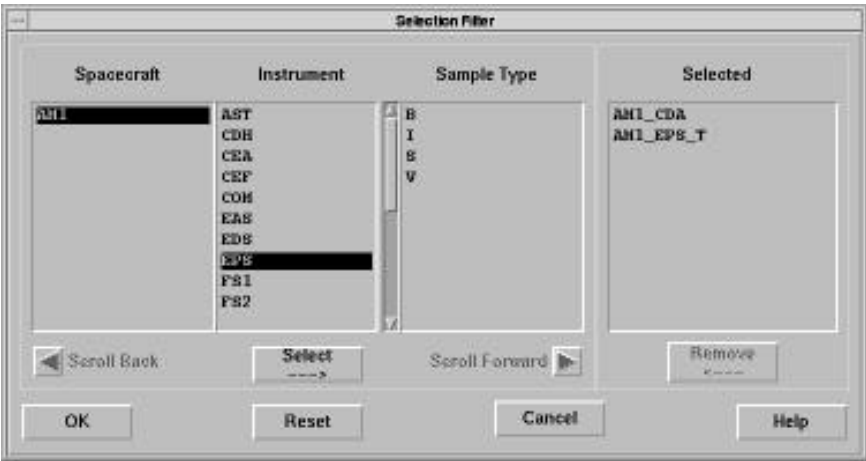


Figure 10.2.1-4. Selection Filter Window

7. Click OK to accept the selected subsystem mnemonics to be used in selecting parameters. The selected subsystem mnemonics are now displayed in the list box labeled Subsystems in the Analysis Telemetry Selector window.
- c. On the Analysis Telemetry Selector window, click on a mnemonic (i.e., AM1_GNC_I) in the list box Subsystems. A list of available parameters for this subsystem mnemonic is displayed in the listbox labeled Available Parameters.
- d. Click a parameter (i.e., a GNC_IR_ACEA_MTR_X) in the list box Available Parameters.
- e. Select a sampling rate by clicking a radio button (i.e., changes only) inside the frame labeled Sampling Rate.

- f. If desired, select a statistics rate by entering an integer in the Statistics text field (e.g., 6). Choose a rate in the pulldown Option menu (e.g., Min).
 - g. Click Select to select parameter GNC_IR_ACEA_MTR_X with sampling rate Changes Only and statistics rate 6 minutes (or 360 seconds). These values are now selected and displayed in the table labeled Selected Parameters.
 - h. Repeat steps 4 through 7 to select additional parameters.
 - i. Click OK to accept the selected parameter(s) with sampling and statistics rates to be used in the Analysis Request. These are now displayed in the table labeled Selected Telemetry in the Analysis Request Builder window.
8. On the Analysis Request Builder window, enter an output data set name in the textfield labeled Output DataSetName; click the toggle button to write the results of this request into this file.
 9. Save this request by clicking the menu option Save As under the menu option File located on the top menu bar.
 10. Click OK to submit this request.

The Analysis Request Builder window is now closed. To modify this request in the Analysis Request window, select **Open** option from the File menu option. A file selection dialog will prompt users to select a file. Enter a file name and click OK. The Save Analysis Request will be in the Analysis Request window.

Each Analysis Request will result in the creation of one or more logical strings. Users should not shutdown user stations with active logical strings associated with Analysis Requests. Please refer to Section 9.2 for more information on logical strings.

10.2.2 Adding User-Defined Algorithms to an Analysis Request

User-Defined Algorithms are C/C++ algorithms written by users and dynamically linked to the analysis request process. User algorithms can take up to twenty (20) input parameters and output up to twenty (20) parameters into the dataset generated by the analysis request. These output parameters can be placed in reports, tables, and graphs like other parameters.

Since FOS software can run on multiple platforms, and analysis requests can be farmed out to any of these platforms, the process of compiling user-defined algorithms will be a configuration controlled procedure, requiring platform specific compiler flags.

- a. **Sample User-Defined Algorithm written in C++, with explanatory comments:**

```
#include "EcTypes.h" // include file used to define User Algorithm specific data types
/* the following variables map to input and output parameters (mnemonics) of the algorithm and
MUST be
global */
/*INPUTS #a required comment needed by the User Algorithm Process. Distinguishes input and
output
parameters */
EcTReal SolarArrayCurrent1; // an input. EcTReal is an algorithm data type for floating point
numbers
EcTReal SolarArrayCurrent2; // another input
```

```

EcTInt SolarArrayCount; // and integer data type input
//OUTPUTS # another required comment, indicating the remaining variables map to outputs
EcTReal TotalCurrent; // a floating point output
EcTInt AnInteger; // and integer output
EcTBoolean AnInteger_OutputComplete = EcTFalse; /* an output flag telling the algorithm software
whether to output the value of AnInteger. Useful for performing running sums where the algorithm
may execute every second but results are only needed every ten minutes. */
extern "C" UserFunction() /* algorithms must be extern "C", return void, and take no arguments.
                           The function name must be the same as the filename minus
                           extension ie. UserFunction.C would be the filename for this
                           algorithm */

{
static int count = 0; // local variables need not be EcTInt/EcTReal
// compute total solar array current and store in output
TotalCurrent = SolarArrayCurrent1 + SolarArrayCurrent2;
// now perform a meaningless example of a running sum
count += SolarArrayCount;
if (count > SOME_BIG_NUMBER)
{
    AnInteger = count;
    AnInteger_OutputComplete = EcDTrue; /* tell the software to output AnInteger */
    count = 0;
}
else
{
    AnInteger_OutputComplete = EcDFalse; /* tell the software NOT to output AnInteger */
}
}

```

- b. **User-Defined Algorithm Registration.** Once a user-defined algorithm is compiled and copied to the appropriate CM area (operational, test, or training) it must be registered. This involves opening the Algorithm Registration window (see Figure 10.2.2-1) from the Tools menu, selecting the appropriate algorithm, and clicking **Register**. Selecting an algorithm can be accomplished by typing the name of the algorithm into the Source File Name text field or by clicking **Select File** and selecting an algorithm name from the file selection menu. FOS software will verify the format of the algorithm source file and report either success or a list of errors found.
- c. **Adding User-Defined Algorithms to an Analysis Request.** When adding algorithms to an Analysis Request, you select **Algorithms** from the Analysis Request Builder to open the Algorithm Request window (see Figure 10.2.2-2). From there, you select the algorithm to be added. At this point you must make several selections before clicking **OK**.

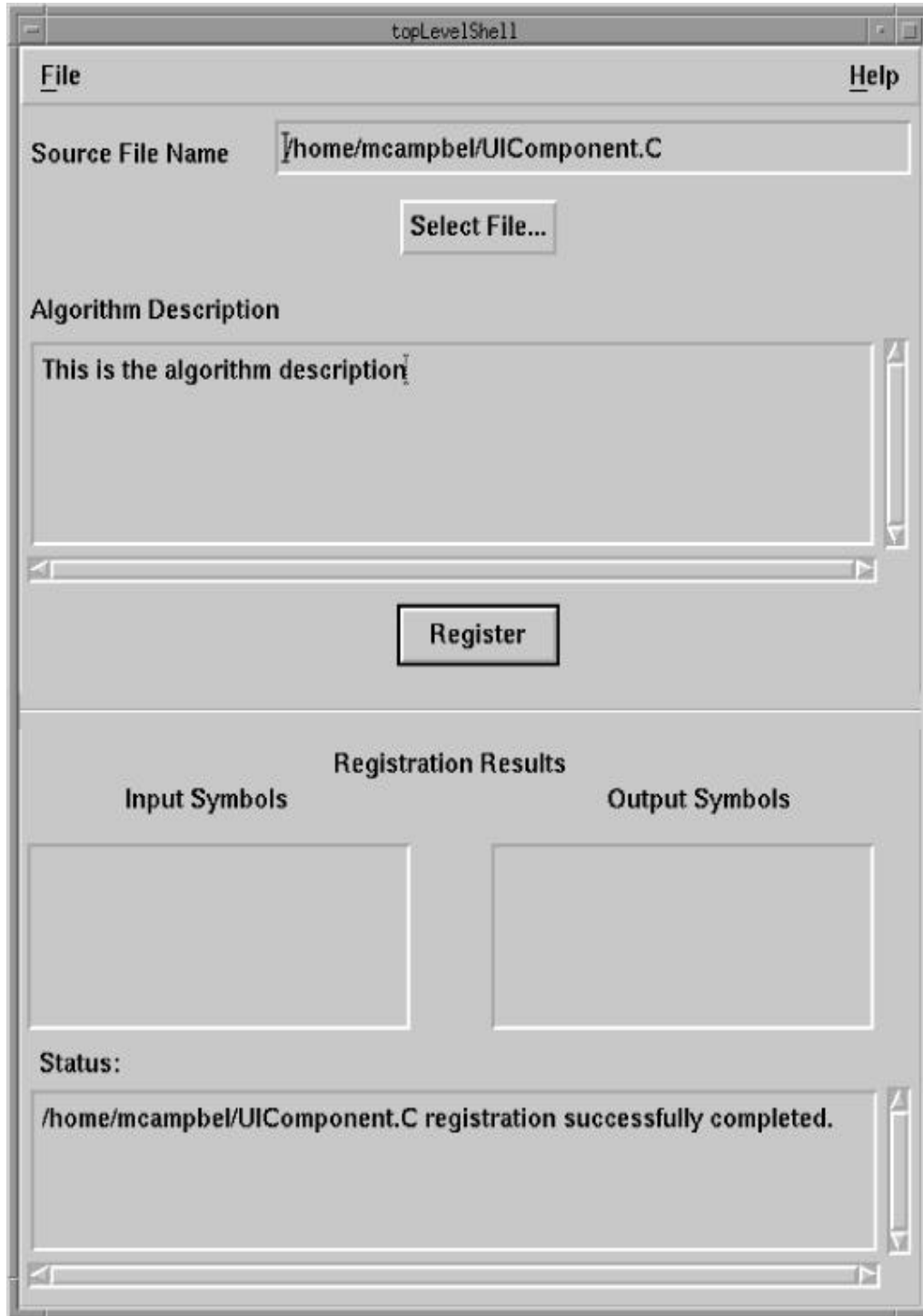


Figure 10.2.2-1. Algorithm Registration Window

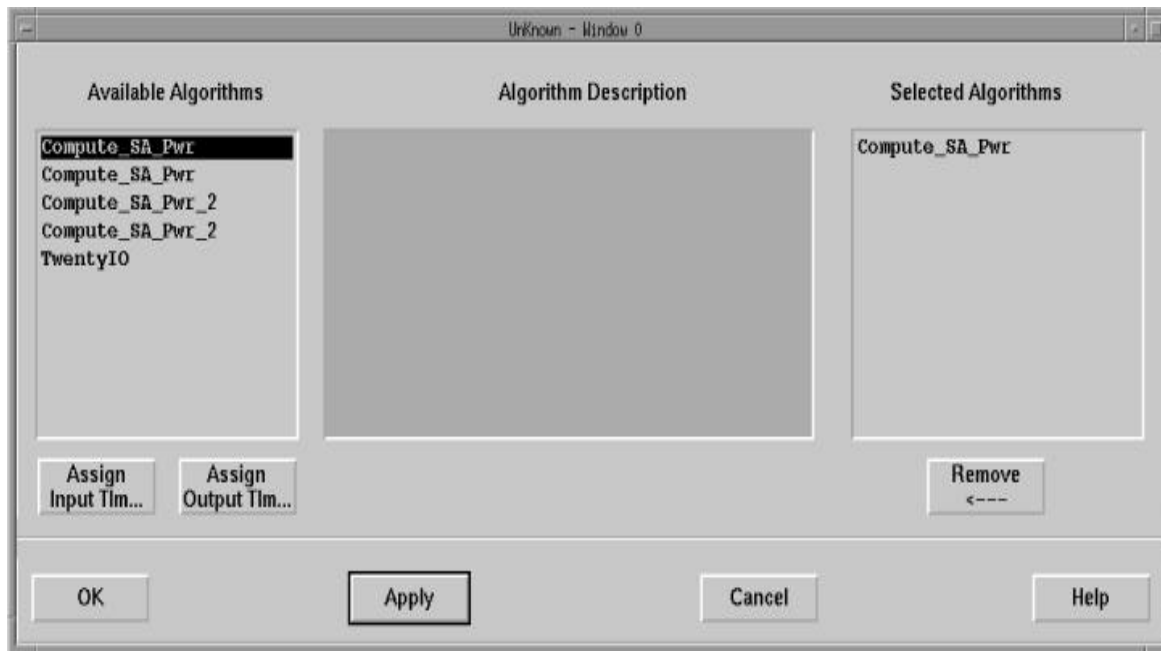


Figure 10.2.2-2. Algorithm Request Window

An algorithm invocation timing mechanism must be chosen. This will determine exactly when and how often the algorithm is invoked. You may select a timer value, that will cause the algorithm to be invoked every N seconds, or select telemetry parameters that trigger the algorithm execution. All parameters selected as triggers will update before the algorithm is invoked. Parameters selected as triggers need not be part of the algorithm. For example, the packet ID parameter could be used as a trigger parameter to cause the algorithm to update once every packet, even though the packet ID parameter is not used in the algorithm.

Selecting the invocation timing mechanism is accomplished by clicking **Assign Input Tlm** at the Algorithm Request window. This will invoke the Assign Input Telemetry window that allows you to specify the timing rate, parameters, and triggers.

The input and output symbols in the algorithm source file must be mapped to input telemetry parameters and output parameters. This is done by clicking **Assign Input Tlm** and **Assign Output Tlm**, respectively. Once the mapping screens are available (see Figures 10.2.2-3 and 10.2.2-4), a selection of Input/Output symbols are shown, along with a telemetry selector for choosing telemetry to map to each Input/Output symbol. When all Inputs and Outputs are mapped, the algorithm selection process is complete.

Upon clicking **OK** at the Assign Input Tlm and Assign Output Tlm windows, you **MUST** click **Apply** from the Algorithm Request window to associate the selected inputs and outputs with the selected algorithm.

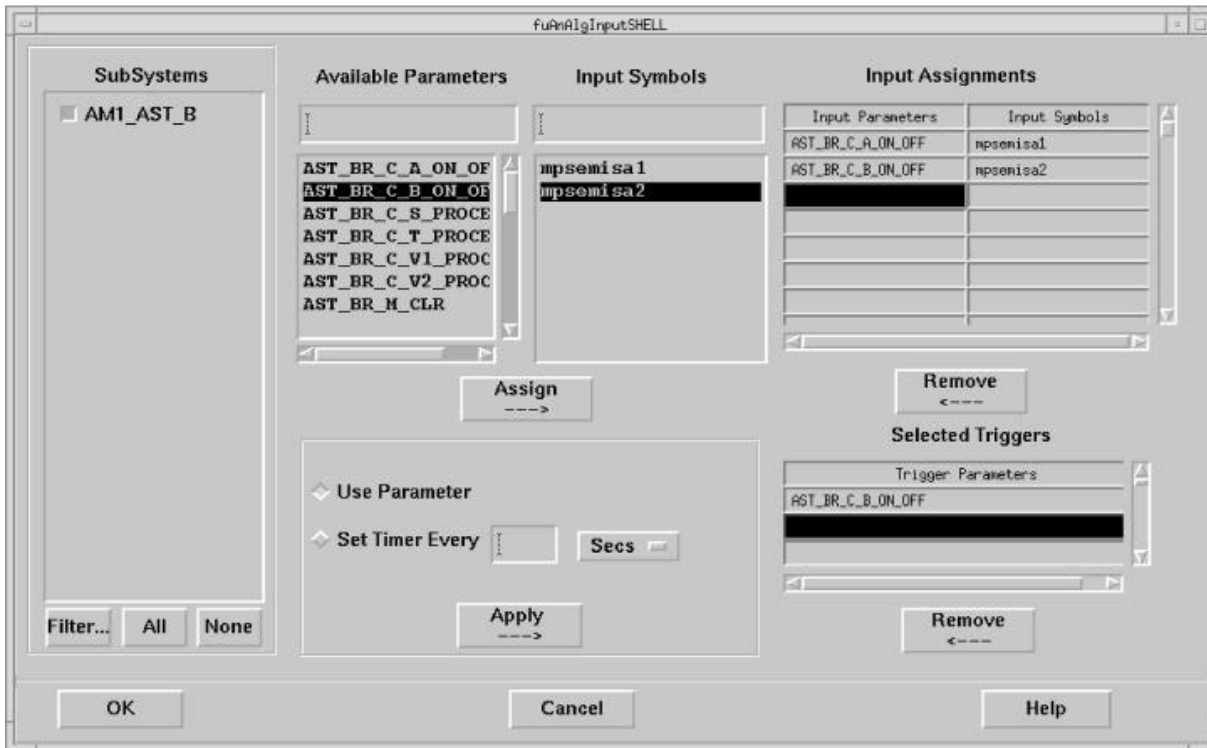


Figure 10.2.2-3. Algorithm Input Parameter Mapping Window

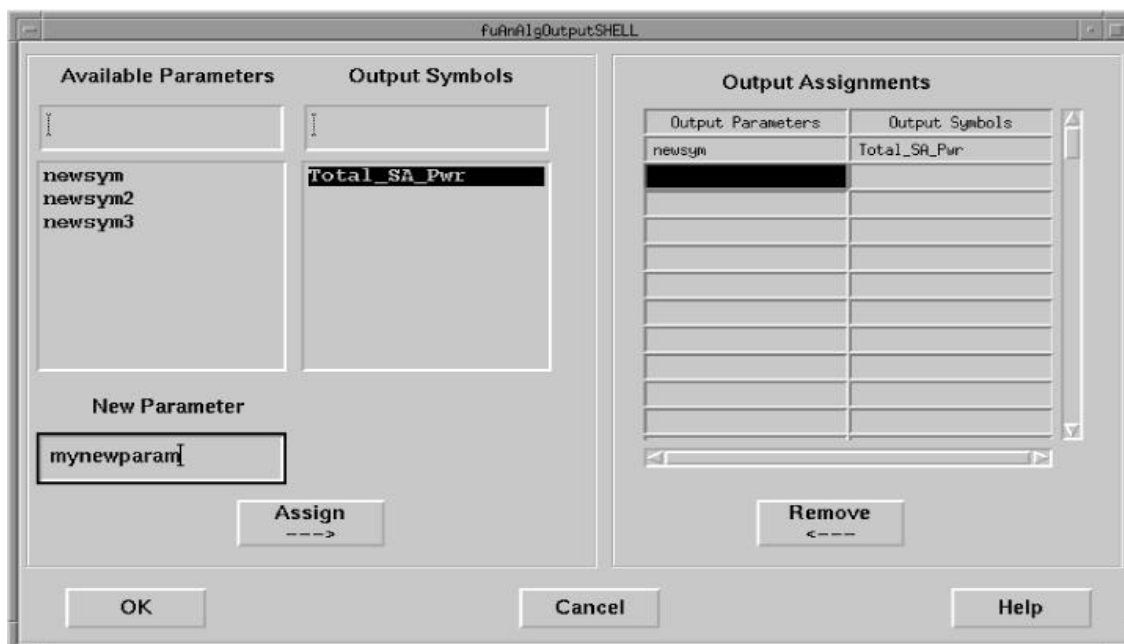


Figure 10.2.2-4. Algorithm Output Parameter Mapping Window

After selections have been applied, the algorithm name will appear in the Selected Algorithm list box on the Algorithm Request window. This allows you to re-select the algorithm for modifications to the input/output mapping or the invocation rate. Algorithms in the Selected Algorithms list can also be selected for deletion. This is done by clicking **Remove** on the Algorithm Request window.

10.3 History Processing

10.3.1 Spacecraft Telemetry

AM-1 spacecraft telemetry data types include housekeeping, health and safety, and standby. Telemetry can be replayed in shared or dedicated fashion and is used to support Analysis Request processing. Telemetry data are stored in hourly files using the following naming convention:

spacecraft + spacecraft time + "." + data type + channel

where:

spacecraft = AM1.

spacecraft time = Spacecraft time, Julian format to the hour (yyyydddh).

data type = HK, HS, SB.

channel = or Q (Applies to a real-time contact only; Does not apply to rate-buffered data).

So, the file AM1199610010.HK1 would represent the I-channel, real-time AM-1 housekeeping data for spacecraft time of: year 1996, day 100, and hour 10. These telemetry files are located in the directory build as follows:

TLM_Path

This directory is an environment variable that can be reconfigured by the Database Administrator. For example: /net/beeper/fosa/dev/AM1

10.3.2 Ground Telemetry

Ground telemetry data types include NCC user performance data, NCC GCM buffers, and EDOS CODAs. The data are stored in files named as follows:

spacecraft + ground time + "." + data type

where:

spacecraft = AM1

ground time = ground receipt time for NCC buffers, embedded time for EDOS CODAs

data type = GCM, UPD, EDOS

So, the file AM1199830015.EDOS would contain the EDOS CODAs received during 1998, day 300, and hour 15 for AM-1. The files are located in the same directory as the files for spacecraft telemetry (refer to Section 10.2.1).

- a. **NCC Performance.** NCC User Performance Data can be replayed or processed in the same way as spacecraft telemetry data.
- b. **EDOS CODA.** EDOS CODAs can be replayed or processed in the same way as spacecraft telemetry data.

10.3.3 Command

- a. **CLTUs.** Analysis of Command Link Transmission Units (CLTU) will be available in Release B.
- b. **CLCWs.** Analysis of Command Link Control Words (CLCW) will be available in Release B.

10.4 Database Management

The FOS Data Management Subsystem (DMS) uses a database to store and manipulate data to support ECS mission operations. A Sybase server provides the foundation for the FOS Database. The installation of the FOS Database is described in the upcoming FOS EOC Installation Guide.

A member of the FOT will be designated as the Database Administrator (DBA). The DBA will have prime responsibility for overseeing the FOS Database and its related operations. A user interface for database operations is available through WWW windows. The DBA has access to the options shown on the Database Utilities window shown in Figure 10.4-1. All other FOT members have access to options shown on the Database Utilities (User) window shown in Figure 10.4-2.

10.4.1 Database Administration

The FOS DBA accesses the FOS Database through the *fos_dba* Sybase account. This account owns all objects within the FOS Database and therefore has all privileges to the data. The DBA's .cshrc file sources two Unix scripts to set environment variables to access the FOS Database.

The *fos_sybsetup.script* sets the Sybase parameters and other environment variables used to interface to Sybase. This file is modifiable only by the System and/or Database Administrators. Any users accessing the FOS Database need to source this script in their .cshrc file.

The *fos_dba_env.script* sets the environment variables for the DBA account. This file is readable/modifiable only by the System and/or Database Administrators. Whenever the administrator modifies the *fos_dba* Sybase account password, a modification to this file is required.

10.4.2 PDB Processing

The PDB generation process is performed by the DBA in a non-real-time, interactive environment. This process provides definitions needed to support mission operations. To ensure successful processing, you should verify proper configuration of the environment. The environment variable, \$PDB_DIR, must be configured to the PDB operational directory. All database scripts must be run from their designated directory for the AM1 mission (e.g., /FOS/OPS/AM1/DB).

The PDB generation process begins upon notification of the transfer of the PDB files to the dedicated area, \$PDB_DIR/input, at the EOC. These files are derived from the most recent version of the Integration & Test (I&T) database supplied by the spacecraft contractor. These definition files, which contain telemetry and command information, will be loaded into the FOS Database \$UNV_DB, by clicking **PDB load** on the FOS Database Utilities window. Each new release of the I&T database supersedes the previous version. During this process, input files are moved to a newly created subdirectory and named according to the I&T version of the input files.

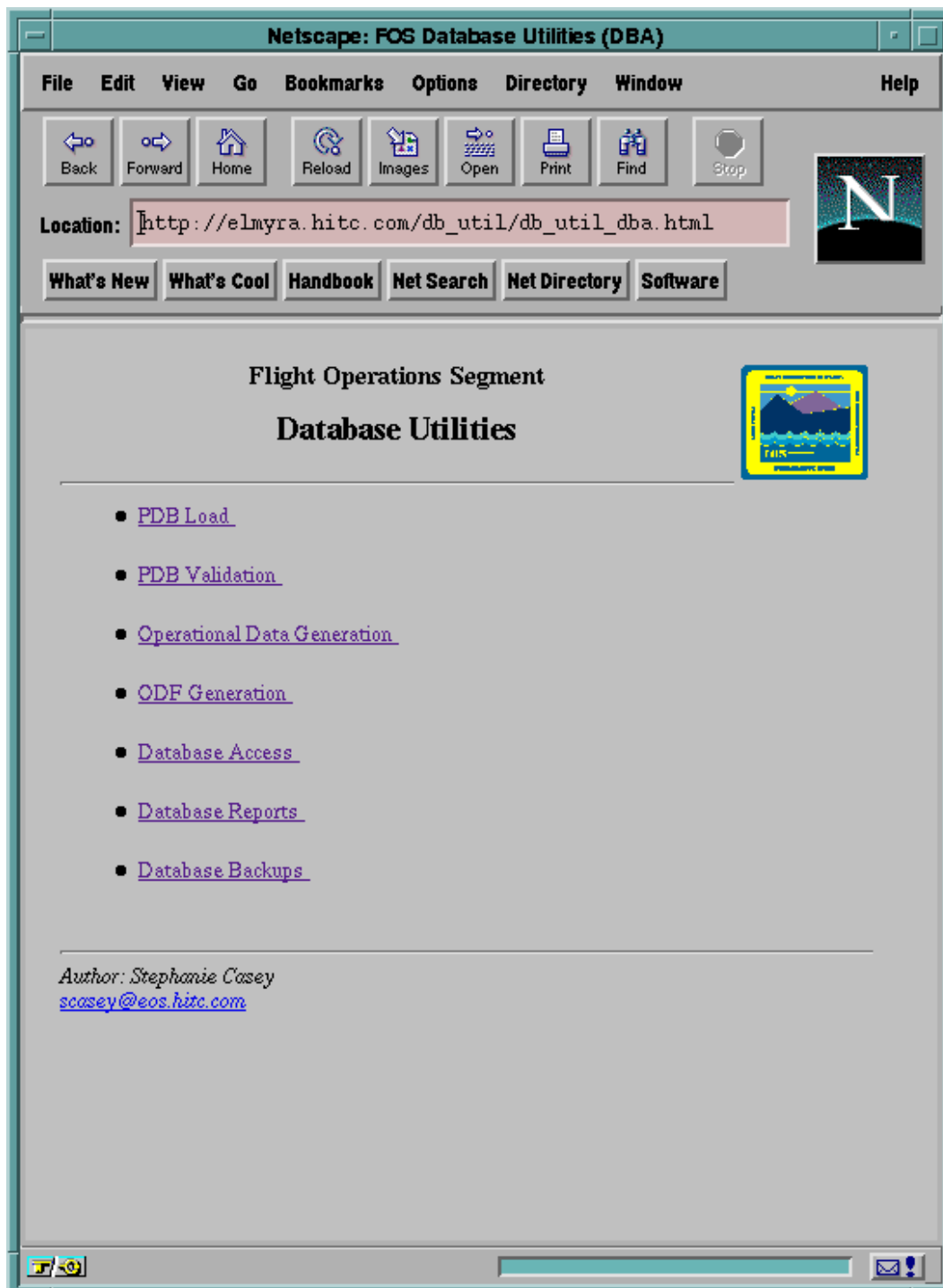


Figure 10.4-1. Database Utilities (DBA) Window

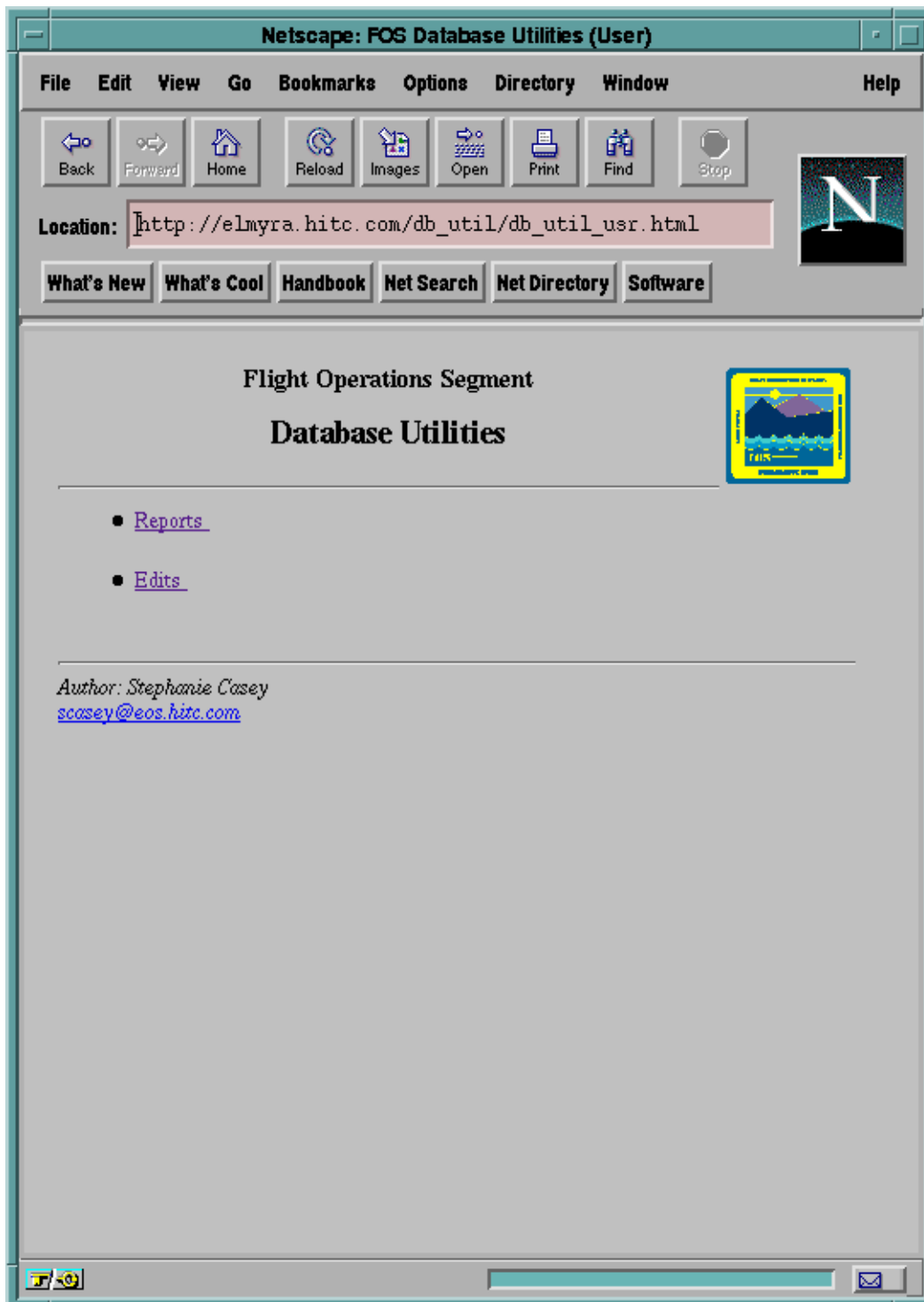


Figure 10.4-2. Database Utilities (User) Window

Validation of PDB definitions ensures the accuracy of information used to support mission operations. It is a necessary step which includes syntax checking, verification of values, and cross-checking of related definitions. Additionally, due to dependencies between PDB data types (i.e., command definitions contain telemetry parameters), a specific order must be enforced when validating these definitions. PDB validation is invoked by clicking **PDB Validation** on the FOS Database Utilities window. The validation options are displayed on the PDB Validation window (see Figure 10.4.2-1).

During the validation process, data is moved into the valid database within the FOS database. Upon completion, a validation summary is available for the DBA's review by clicking **View the PDB Validation Output File** on the PDB Validation window.

Operational database tables are generated by clicking **Operational Data Generation** on the FOS Database Utilities window. Operational data generation options include generating a new operational database, generating a new version of constraints only, or new activities only (see Figure 10.4.2-2). Valid information is moved from the valid database into the operational database within the FOS database and a set of dump files are created.

The operational database file generation process creates the input files used by various FOS subsystems. Each ODF build operation creates one or more output files in the output directory \$ODF_DIR (refer to Table 10.4.2-1).

Table 10.4.2-1. ODF Source Generation

EXECUTABLE	ODF
FdDbBuildAnlOdf	AnalysisOdf_\$ODFVERSION
FdDbBuildCmsOdf	CommandOdf_\$ODFVERSION
FdDbBuildEventOdf	EventODF_\$ODFVERSION
FdDbBuildFuiCmdOdf	FuiCommandODF_\$ODFVERSION
FdDbBuildFuiOdf	ParmDataOdf_\$ODFVERSION
FdDbBuildNccOdf	GcmrOdf_\$ODFVERSION
FdDbBuildPidsOdf	TlmSelectFilterOdf_\$ODFVERSION
	CmdSelectFilterOdf_\$ODFVERSION
	PidsOdf_\$ODFVERSION
FdDbBuildSysOdf	SystemOdf_\$ODFVERSION
FdDbBuildTlmOdf	TlmDecom Odf_\$OdfVersion
	TlmDumpOdf_\$OdfVersion
FdDbBuildActivityOdf	ActivityOdf_\$ODFVERSION
FdDbBuildCmdConOdf	CmdConOdf_\$OdfVersion
FdDbBuildCmdFormatOdf	CmdFormatOdf_\$OdfVersion
FdDbBuildCodaOdf	CodaOdf_\$OdfVersion
FdDbBuildDetStringOdf	DetStringOdf_\$OdfVersion
FdDbBuildFuiTlmMnemOdf	FuiTlmMnemOdf_\$OdfVersion
FdDbBuildMemoryMaskOdf	MemoryMask_\$OdfVersion
FdDbBuildMnemToPidOdf	MnemToPidOdf_\$OdfVersion
FdDbBuildUpdOdf	UpdOdf_\$OdfVersion



Figure 10.4.2-1. PDB Validation Window

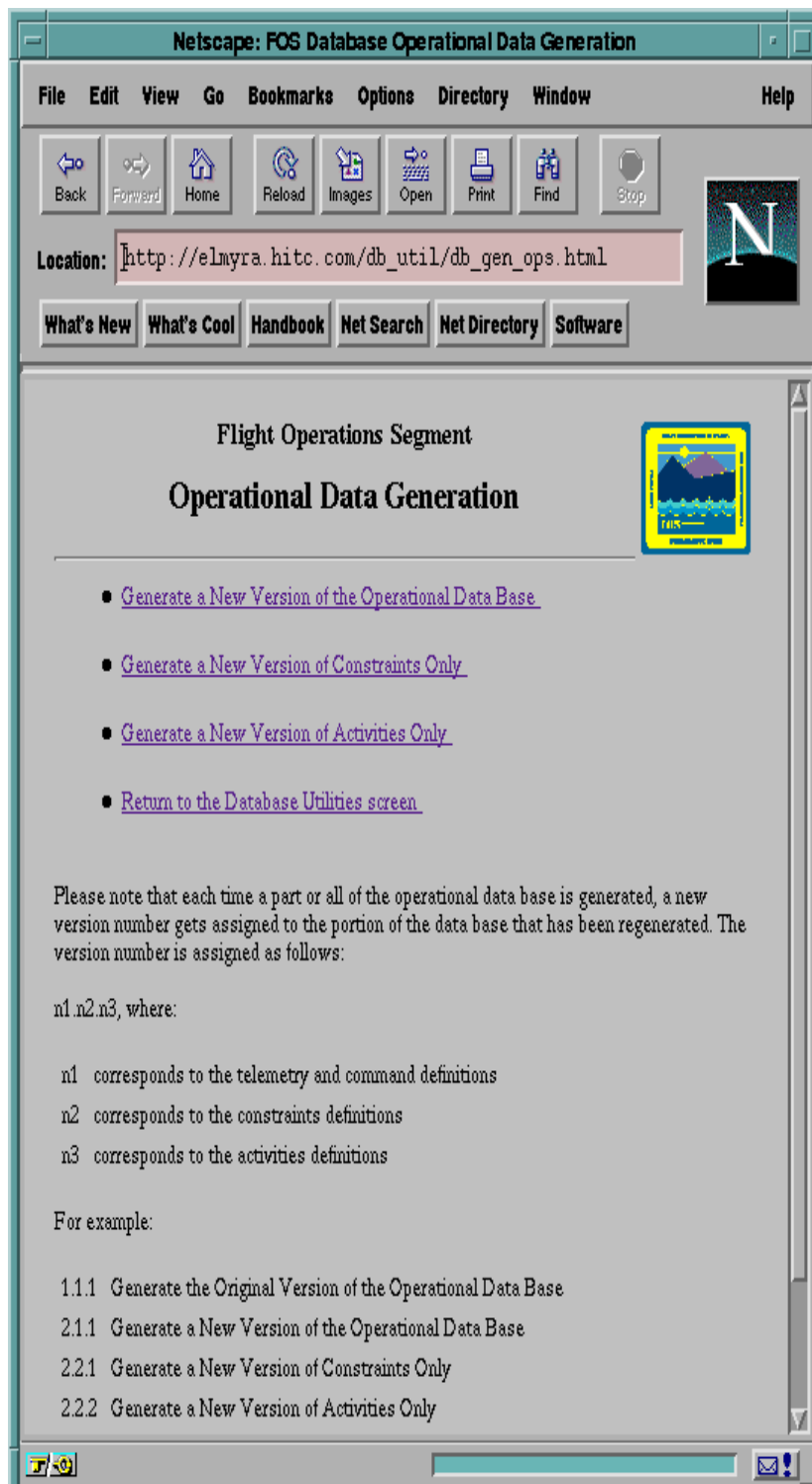


Figure 10.4.2-2. Operational Data Generation Window

10.4.3 FDF Products Processing

The FDF generation process is performed in a non-real-time, interactive environment. This process provides definitions needed to support mission operations. To ensure successful processing, you should verify proper configuration of the environment. The environment variable \$FDF_DIR must be configured to the FDF operational directory. All database scripts must be run from their designated directory, such as /FOS/OPS/AM1/DB for AM-1.

The FDF generation process begins upon notification of the transfer of the FDF files to the dedicated directory, \$FDF_DIR/input, at the EOC. All files transferred between FDD and EOC are conducted through the Kerberos File Transfer Protocol (KFTP). The FDF product files will be loaded into the FOS Database \$UNV_DB. When File Watcher encounters a signal file from FDD, the database script, fdf_load.script, will be invoked. Each new release of the I&T database is appended to the existing database. During this process, input files are moved to a newly created subdirectory that reflects the time the files were received.

For example:

FDF original product file input/AM1_FDFRANGE1_017_1991_1.FDD

File after load data/am1_range112430.data

Files sent from FDF containing errors will be batched and returned to FDF for manual editing. No online editing capabilities will be provided.

Validation of FDF definitions ensures the accuracy of information used to support mission operations. It is a necessary step which includes syntax checking and verifying of values for both the header and data files.

The validation process is triggered by the File Watcher. When an FDF signal file is encountered, File Watcher also invokes, exec_valdata_proc.script, and fdf_header_proc.script.

Reports for each FDF product will be available by clicking **Database Reports** on the Database Utilities window (refer to Section 10.4).

Generation of operational database tables is invoked through the database script, gen_fdf_ops_db.script. Valid information is moved from the valid database, \$VAL_DB, into the operational database, \$OPS_DB, within the FOS database.

10.4.4 Database Access

The database access function, allowing you to access the FOS database, is opened by clicking **Database Access** on the FOS Database Utilities window. The options are displayed on the Database Access window (see Figure 10.4.4-1).

The Database Access windows provide you with the ability to manipulate data in the database through Netscape.

At the bottom of the database access forms is a **Clear Form** button that clears all previous entries and a **Submit** button that sends the request to the database. You may submit a request with any number of the fields entered. This will filter the search and return records that meet each of the field specifications. A form submitted without any entries will return all records in the database.



Figure 10.4.4-1. Database Access Window

10.4.4.1 Activity Log

The Spacecraft Activity Log Monitor process is begun when a string is started. The process will monitor telemetry for activity log messages. For each new activity log message received, an event is created. The event indicates whether the activity log was severe or informational, the activity log entry ID, the message associated with that ID, the time the entry was created, the cycle, and the five data words for the message. The Activity Log Monitor process also receives the dump data when the Activity Log table is dumped. When the monitor finishes processing the dump, one event

message will be generated containing the number of new activity log messages and how many of those were severe.

All activity log messages are archived to the database and can be viewed through the Activity Log Web Browser.

Open the Activity Log Database Access window (see Figure 10.4.4.1-1) by clicking **Activity Log** on the Database Access window. The Activity Log window is used to retrieve spacecraft activities from the activity log archive.

Netscape: Activity Log Database Access Form

File Edit View Go Bookmarks Options Directory Window Help

Back Forward Home Reload Images Open Print Find Stop

Location:

What's New What's Cool Handbook Net Search Net Directory Software

Flight Operations Segment

Activity Log Database

Activity ID # Source

Spacecraft Time Stamp: (to)
(format YYYY:DDD:HH:MM:SS)

Log Position

Major Cycle/ Minor Cycle

FOS Activity Message

Clear Form

Submit

Author: Nick Lutzio
nlutzio@eos.hitc.com

Figure 10.4.4.1-1. Activity Log Window

The following is a list of fields associated with the activity log along with a brief field description.

- Activity ID#.** Each activity has an associated identification number. Possible values for this field are 0 to 256.

- b. **Source.** The source of data can either be real-time or simulation. This field also has * as an option. The asterisk will select activities from both sources.
- c. **Spacecraft Time Stamp.** You have the ability to retrieve activities within a certain Time Stamp range. The format for this field is: yyyy.ddd.hh.mm.ss.
- d. **Log Position.** The spacecraft contains 300 log positions to store activities before they are dumped. This field allows you to select which log position to filter on.
- e. **Major Cycle.** Activities are trickled down with telemetry during each support.
- f. **The Minor Cycle.** You can select the major/minor cycle with which the activity trickled down.
- g. **FOS Activity Message.** This field is used to retrieve any activities containing the string of characters entered into this field.

10.4.4.2 FOS Database Events

Open the FOS Database Events window (see Figure 10.4.4.2-1) by clicking **FOS Events** on the Database Access window.

- A. **Event Definitions.** Open the Event Definitions window (see Figure 10.4.4.2-2) by clicking **Event Definitions** on the FOS Database Events window.

The Event Definitions window is used to define and view information stored in the Event ODF. The following items are a list of fields associated with the event definitions along with a brief field description:

1. **Event Type.** The FOS subsystem that generated the event: ANA, CMS, CMD, DMS, FUI, PAS, RCM, RMS, TLM, or SYS.
2. **Severity.** The severity of the event. Valid severities are Fatal, Alarm, Warning, and Informational.
3. **Definition ID.** The event as a number. All integers are valid for this field.
4. **Event Name.** The field as a string. This field should follow the FOS naming convention for events (2-character subsystem ID, "CEv", Descriptive Name). An example of a DMS event is, FdCEvInvalidUserRequest.
5. **Background Text.** The background text stored in the Events Definitions file. This field should contain %s where a string needs substitution, a %d where an integer needs substitution, and a %t where time needs substitution.

An example of a background text is "Printer %s Failed". The %s is filled by the application generating the event.
6. **Trigger.** The name of a procedure or executable associated with an event. For most events this field will be blank.
7. **Process Routing Name.** The name of the process event that needs to be routed. For most events this field will be blank.



Figure 10.4.4.2-1. FOS Database Events Window

Netscape: Event Definitions

File Edit View Go Bookmarks Options Directory Window Help

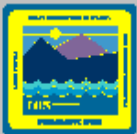
Back Forward Home Reload Images Open Print Find Stop

Location:

What's New What's Cool Handbook Net Search Net Directory Software

Flight Operations Segment

Event Definitions



Event Definition Fields (empty fields are wild-carded)

Event Type : Severity

Definition ID # (for definition, find, remove, or update only)

Event Name

Background Text

(Event Trigger)

(Processing Route)

Figure 10.4.4.2-2. Event Definitions Window

- B. **Event History.** Open the FOS Event History window (see Figure 10.4.4.2-3) by clicking **FOS Events** on the FOS Database Events window.

The screenshot shows a Netscape browser window titled "Netscape: Event History". The address bar contains the URL "http://elmyra.hitc.com/evhistform.html". The browser's menu bar includes File, Edit, View, Go, Bookmarks, Options, Directory, Window, and Help. Below the menu bar is a toolbar with icons for Back, Forward, Home, Reload, Images, Open, Print, Find, and Stop. A "Location:" field is present, followed by buttons for "What's New", "What's Cool", "Handbook", "Net Search", "Net Directory", and "Software".

The main content area is titled "Flight Operations Segment" and "Event History". It contains several search filters:

- Spacecraft** and **Subsystem**: Each has a dropdown menu with an asterisk (*) and a small arrow icon.
- Spacecraft Time Stamp**: Two text input fields separated by "(to)". Below them is the format "(format YYYY:DDD:HH:MM:SS)".
- FOS Event Type** and **Severity**: Each has a dropdown menu with an asterisk (*) and a small arrow icon.
- FOS Time Stamp**: Two text input fields separated by "(to)". Below them is the format "(format YYYY:DDD:HH:MM:SS)".
- FOS Event ID #**: A single text input field.
- FOS Event Message**: A single text input field.
- FOS Trigger**: A single text input field.

At the bottom of the window, there is a status bar with a small icon on the left and a progress bar on the right.

Figure 10.4.4.2-3. FOS Event History Window

The Event History window allows you to retrieve events from the event archive. The following is a list of fields associated with the event history along with a brief field description:

1. **Spacecraft.** Currently, AM1 is the only valid spacecraft.

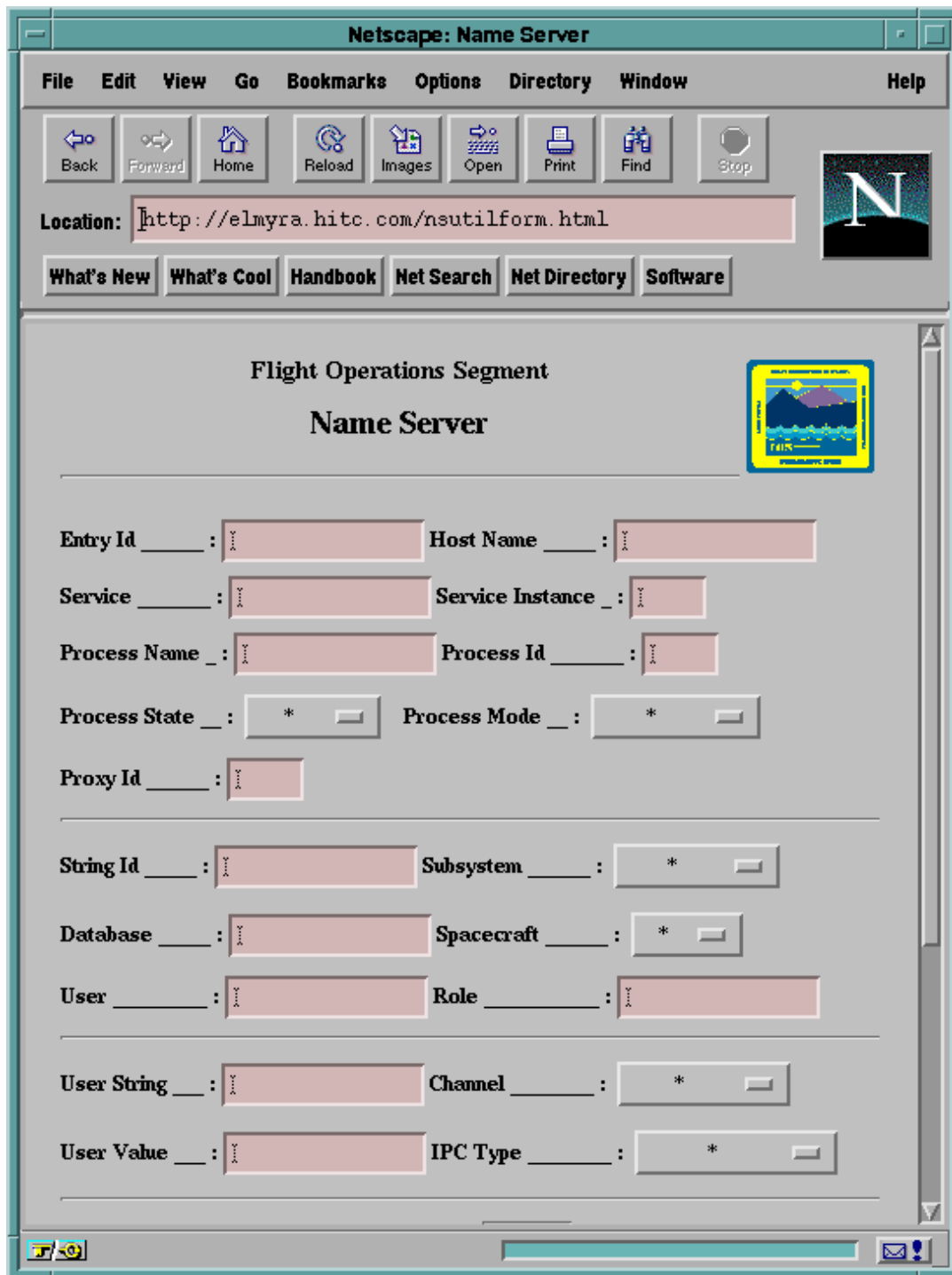
2. **Subsystem.** The spacecraft or ground subsystem. Examples are Aster, Modis, FDF, and NCC.
3. **Spacecraft Time Stamp.** The start and stop spacecraft times of the event request.
4. **Event Type:** The FOS subsystem generating the event. Valid subsystems are ANA, CMS, CMD, DMS, FUI, PAS, RCM, RMS, TLM, and SYS.
5. **Severity.** The severity of an event. Valid severities are Fatal, Alarm, Warning, and Informational.
6. **FOS Time Stamp.** The start and stop UTC times of the event request
7. **Definition Id.** The event identifier.
8. **FOS Event Message.** The event message displayed on the Event Display window.
9. **Trigger.** The name of the procedure or executable associated with the event.
10. **FOS Host.** The host name where an event was generated.
11. **Source File.** The source file generating the event.

10.4.4.3 Name Server

Open the Name Server window (see Figure 10.4.4.3-1) by clicking **Name Server** on the Database Access window. The Name Server window allows you to access the Name server database. The following is a list of fields associated with the Name server along with a brief field description:

1. **Entry Id.** The port number of the process.
2. **Host Name.** The host where the process is running.
3. **Service.** The service the process provides.
4. **Service Instance.** The instance of the process on a given host.
5. **Process Name.** The name of the process.
6. **Process Id.** The Unix process ID.
7. **Process State.** The state of the process. Valid states are Active, Backup, and Inactive.
8. **Process Mode.** The mode of the process. Valid modes are Operational, Test, and Training.
9. **Proxy Id.** The class ID of a proxy.
10. **String Id.** The process is associated with this Decom and Parameter Server String. The field is blank if the process is not associated with a string.
11. **Subsystem.** The FOS Subsystem. Valid subsystems are ANA, CMD, CMS, DMS, RCM, RMS, TLM, FUI, and PAS.
12. **Database.** The process is connected to this database. Blank, if not connected to a database.
13. **Spacecraft.** Currently, the only valid spacecraft is AM1.
14. **User.** The user name of the person starting the process.
15. **Role.** The role of the person starting the process.
16. **User String.** The open field.

17. **Channel.** The Channel I = 1, ChannelQ=2, Replay=3, and Config=5.
18. **User Value.** The open field.
19. **IPC Type.** The valid IPC types are Point-to-Point and Multicast.



The screenshot shows a Netscape browser window titled "Netscape: Name Server". The address bar contains "http://elmyra.hitc.com/nsutilform.html". The main content area is titled "Flight Operations Segment Name Server" and contains several input fields and dropdown menus for configuring a name server entry.

Flight Operations Segment Name Server

Entry Id ____ : Host Name ____ :

Service ____ : Service Instance ____ :

Process Name ____ : Process Id ____ :

Process State ____ : Process Mode ____ :

Proxy Id ____ :

String Id ____ : Subsystem ____ :

Database ____ : Spacecraft ____ :

User ____ : Role ____ :

User String ____ : Channel ____ :

User Value ____ : IPC Type ____ :

Figure 10.4.4.3-1. Name Server Window

10.4.4.4 PDB Parameters

Open the PDB Parameters window (see Figure 10.4.4.4-1) by clicking **PDB Parameters** on the Database Access window.

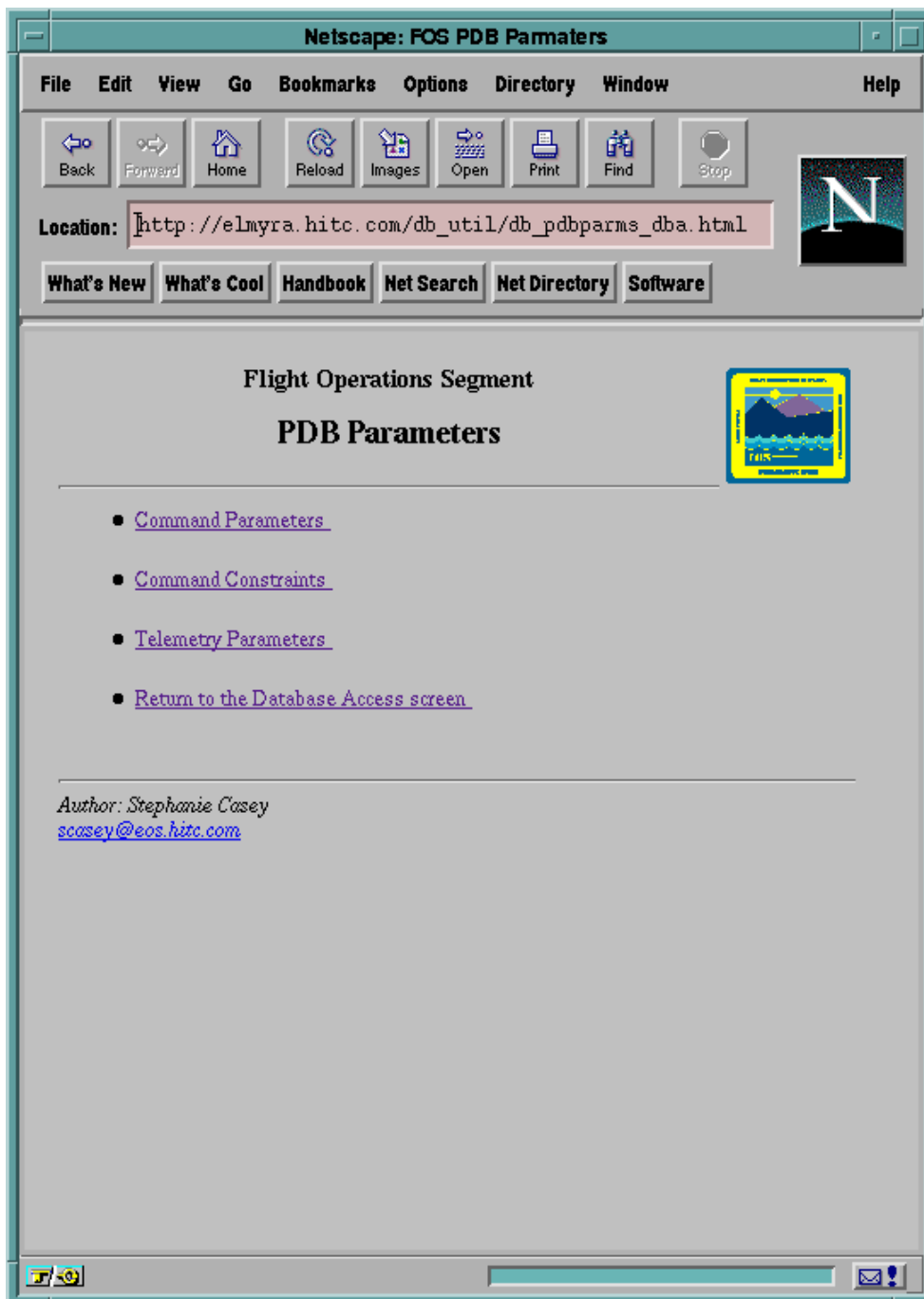


Figure 10.4.4.4-1. PDB Parameters Window

10.4.4.4.1 Command Parameters

Open the Command Parameters window (see Figure 10.4.4.4.1-1) by clicking **Command Parameters** on the PDB Parameters window.

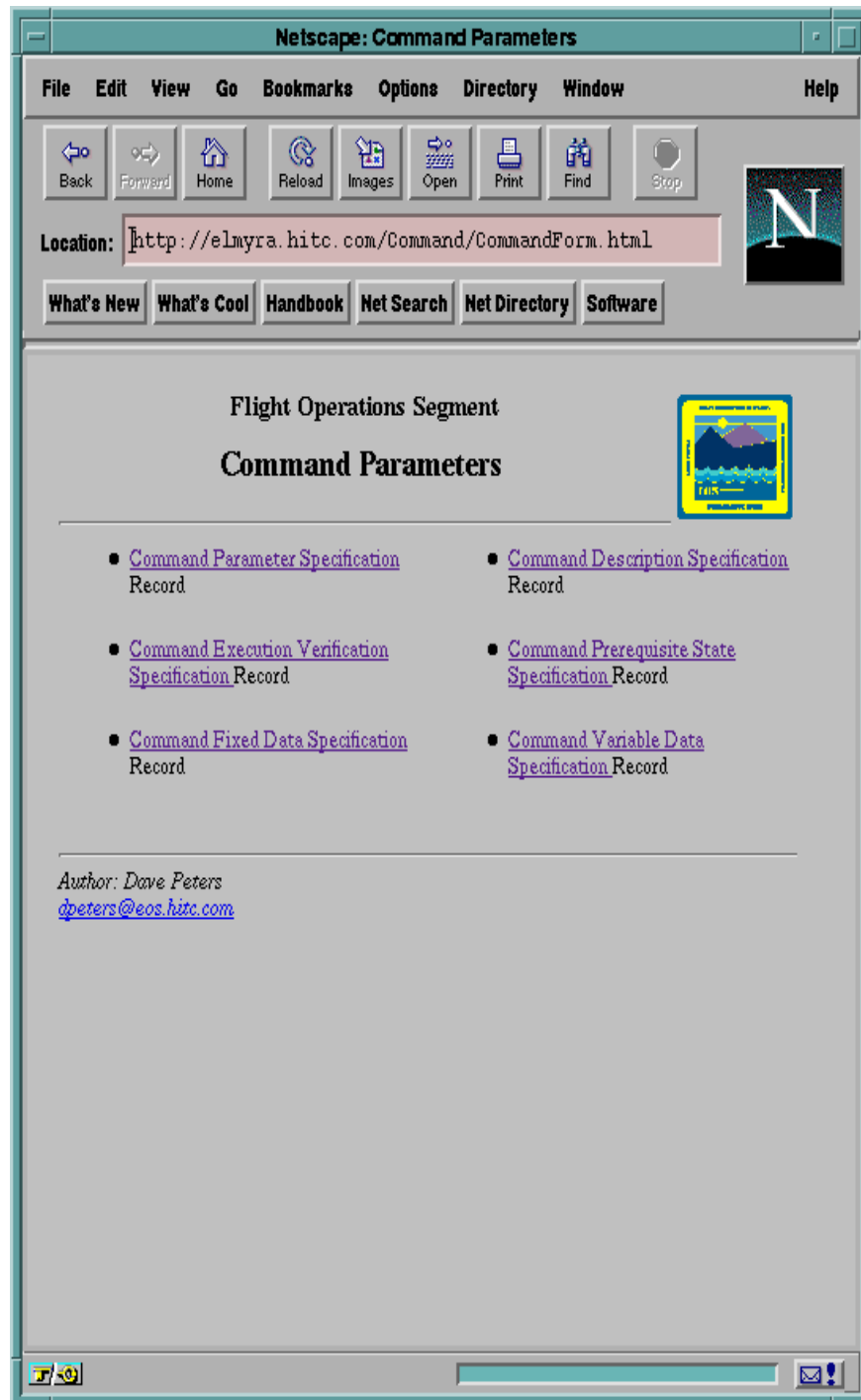


Figure 10.4.4.4.1-1. Command Parameters Window

A. **Command Parameter Specification.** Open the Command Parameter Specification window (see Figure 10.4.4.4.1-2) by clicking **Command Parmeter Specification** on the Command Parameters window. The Command Parameter Specification Record defines a spacecraft or instrument command used to support the EOS spacecraft. Each record provides the construction information for a command. The following is a list of fields associated with the Command Parameter Specification Record:

1. **Command Parameter Identifier.** This number uniquely identifies the spacecraft or instrument command parameter.
2. **Command Mnemonic.** Identifies each spacecraft or instrument command parameter. The command mnemonic is a unique name consisting of 14 to 20 characters representing a spacecraft or instrument command for the EOS spacecraft. The format is as follows:

AAA_<Command Verb>_<Command Name>

where:

AAA is the three uppercase characters representing spacecraft subsystem or instrument;

<Command Verb>: 3 to 9 uppercase characters representing the command verb; valid values for the EOS spacecraft include:

ACTIVATE
ARM
BOOT
CHANGE
CLOSE
DISABLE
DISARM
DRIVE
DUMP
ENABLE
FIRE
FLYBACK
FORCEOFF
FORCEON
GET
HALT
IGNORE
INITIATE
LOAD

MLOAD
MOVE
OPEN
PASS
PERFORM
RESET
SELECT
SET
SLEW
STEP
TOGGLE
TURN_OFF
TURN_ON
USE

<Command Name> specifies 6 to 12 uppercase characters representing the command name describing the function to be performed onboard the spacecraft. The actual length of the command name is dependent on the command verb.

3. **Command Type.** Command classification with respect to the Remote Terminal (RT) and end component interface.
4. **Remote Terminal (RT) ID Name.** The name of the remote terminal to which a command is sent to over the 1553 Command and Telemetry (C&T) bus.
5. **RT subaddress Name.** The name of the remote terminal subaddress the command is sent to.
6. **Command Word Count.** The number of 16-bit words following the command destination (1553 message header) in the command structure. The command descriptor and optional command data words are included in this count. A value greater than 1 indicates command data words exist.
7. **Command Data Word Type.** The source of the command data, where:
F = FIXED; assembled from a fixed bit pattern.
V = VARIABLE; user specified.
A FIXED command data word type would indicate the presence of fixed type command data words associated with the command. A VARIABLE command data word type may also contain a fix part of the bit pattern and therefore would have both fixed and variable type command data words.
8. **Safety Level.** Indicates whether the command is critical to the spacecraft hardware and/or personnel. Valid values include:
H = Hazardous. (Critical)
S = Safe.

E = Exclude. The command is to be excluded from the command process.
O = One time only. Indicates commands that are sent one time only.

Netscape: Command Parameter Specification

File Edit View Go Bookmarks Options Directory Window Help

Back Forward Home Reload Images Open Print Find Stop

Location:

What's New What's Cool Handbook Net Search Net Directory Software

Flight Operations Segment

Command Parameter Specification

◆ Command Parameter

Command Mnemonic: Command Parameter Id:

Command Type: Remote Terminal Id:

Word Count: Data Word Type:

Remote Terminal Subaddress: Safety Level:

Clear Form

Access Method

◆ Find

◆ Add

◆ Update

◆ Remove

Add / Update / Remove Password:

Submit

Author: Dave Peters
dpeters@eos.hitc.com

Figure 10.4.4.1-2. Command Parameter Specification Window

- B. **Command Execution Verification (CEV) Record.** Open the Command Execution Verification (CEV) window (see Figure 10.4.4.1-3) by clicking **Command Execution Verification** on the Command Parameters window.

Netscape: Command Execution Verification Specification

File Edit View Go Bookmarks Options Directory Window Help

Back Forward Home Reload Images Open Print Find Stop

Location:

What's New What's Cool Handbook Net Search Net Directory Software

Flight Operations Segment

Command Execution Verification Specification

◆ **Command Execution Verification**

Command Mnemonic: Command Parameter Id:

CEV Mnemonic: DN/EU Indicator:

CEV Low Value: CEV High Value:

CEV State: CEV Timeout:

CEV Mask:

Access Method

◆ **Find**

◆ Add

◆ Update

◆ Remove

Add / Update / Remove Password:

Author: Dave Peters
dpeters@eos.hitc.com

Figure 10.4.4.1-3. Command Execution Verification Specification Window

The CEV Specification Record defines a telemetry parameter used to verify the reception and execution of an associated command by the spacecraft subsystem or instrument. Each command parameter may specify one analog or discrete telemetry parameter to verify execution. A range of values in which the telemetry parameter must occur is specified to verify command execution. To indicate an exact value for the telemetry parameter, the low value and high value must be equal. The following is a list of fields associated with the Command Execution Verification Record:

1. **Command Parameter Identifier.** The number uniquely identifying the spacecraft or instrument command parameter. This value must be specified in the Command Parameter Specification PDB.
2. **Command Mnemonic.** The name of the command parameter. This value must be specified in the Command Parameter Specification PDB in combination with the command parameter ID.
3. **CEV Mnemonic.** The telemetry parameter whose value verifies the receipt and execution of the command. This name must be specified in the Telemetry Description PDB.
4. **DN/EU Indicator.** The units the CEV mnemonic is defined as (i.e., raw data number or engineering units), where.

DN = raw data number

EU = engineering units

This field is only used for analog telemetry parameters, therefore the telemetry parameter must be of type analog. Additionally, a CEV mnemonic expressed in engineering units must also have an associated definition in the Polynomial Coefficients Specification PDB.

5. **CEV Low Value.** The lowest acceptable value, inclusive, of the CEV mnemonic to verify the command has been properly executed onboard the spacecraft. This value cannot be greater than the CEV high value. The format for this field is determined by the DN/EU indicator. A DN/EU indicator set to DN indicates this value will be defined as a raw data number and contains a decimal integer. A DN/EU indicator set to EU indicates this value will be defined in engineering units and contains a floating-point number.
6. **CEV High Value.** The highest acceptable value, inclusive, of the CEV mnemonic to verify the command has been properly executed onboard the spacecraft. This value cannot be less than the CEV low value. The format for this field is determined by the DN/EU indicator. A DN/EU indicator set to DN indicates this value will be defined as a raw data number and contains a decimal integer. A DN/EU indicator set to EU indicates this value will be defined in engineering units and contains a floating-point number.
7. **CEV Time Out.** The maximum time in seconds for verification of the transmitted command to occur before it is considered failed.
8. **CEV Mask.** The bit pattern in hexadecimal format logically combined with the expected value and the value of the telemetry point used to verify the command. The mask concept provides the capability to have a multi-bit telemetry point serve as the

- CEV mnemonic for a command that will only cause 1 bit of the multi-bit point to change.
9. **CEV State.** The text associated with the CEV mnemonic used to verify command execution. This value will override the use of the expected value when both are specified.
- C. **Command Fixed Data Specification.** Open the Command Fixed Data Specification window (see Figure 10.4.4.4.1-4) by clicking **Command Fixed Data Specification** on the Command Parameters window.

The screenshot shows a Netscape browser window with the title "Netscape: Command Fixed Data Specification". The address bar contains the URL "http://elmyra.hitc.com/Command/CommandFixData.html". The page content is as follows:

Flight Operations Segment

Command Fixed Data Specification

Command Fixed Data

Command Mnemonic: Command Parameter Id:

Word Number: Data Value:

Clear Form

Access Method

☒ Find

☐ Add

☐ Update

☐ Remove

Add / Update / Remove Password:

Submit

Author: Dave Peters
dpeters@eos.hitc.com

Figure 10.4.4.4.1-4. Command Fixed Data Specification Window

The Command Fixed Data Specification Record defines the data words (1-N) associated with the fixed bit pattern of a command. Word 1 would represent the command destination and word 2 represents the command descriptor. The following is a list of fields associated with the Command Fixed Data Word Specification:

1. **Command Parameter Identifier.** The number uniquely identifying the spacecraft or instrument command parameter. This value must be specified in the Command Parameter Specification PDB.
 2. **Command Mnemonic.** The name of the command parameter. This value must also be specified in the Command Parameter Specification PDB in combination with the command parameter ID.
 3. **Word Number.** The order of data words associated with a command. Valid values are 1 to 33.
 4. **Command Data Value.** The fixed bit pattern of the data word.
- D. **Command Description Specification.** Open the Command Description Specification window (see Figure 10.4.4.4.1-5) by clicking **Command Description Specification** on the Command Parameters window.

The Command Description Record provides descriptive information concerning a spacecraft or instrument command parameter. The following is a list of fields associated with the Command Description Record:

1. **Command Identifier.** The number uniquely representing the command parameter. This value must also be specified in the Command Parameter Specification PDB.
2. **Command Mnemonic.** The name of the command parameter. This value must also be specified in the Command Parameter Specification PDB in combination with the command parameter ID.
3. **Major Assembly.** The name of the spacecraft major assembly containing the component receiving the command.
4. **Component Name.** The spacecraft component receiving the command.
5. **Subassembly Name.** The name of the subassembly within the component that will be affected by the command. This value is not required by all commands.
6. **ATC Inhibit ID.** The inhibit group of the command.
7. **Store Command Indicator.** Indicates whether the command is executable from the ATC or RTS buffer, where:
0 = false, not executable.
1 = true, executable.
8. **Pseudo Operations Frequency.** The type of command, where:
0 = normal.
1 = activate RTS.
2 = jump.
3 = no op.
4 = halt.
9. **Command Short Description.** Textual information describing the spacecraft or instrument command.

Netscape: Command Description Specification

File Edit View Go Bookmarks Options Directory Window Help

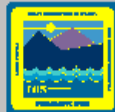
Back Forward Home Reload Images Open Print Find Stop

Location:

What's New What's Cool Handbook Net Search Net Directory Software

Flight Operations Segment

Command Description Specification



◆ **Command Description**

Command Mnemonic:	<input type="text"/>	Command Parameter Id:	<input type="text"/>
Major Assembly:	<input type="text"/>	Subassembly:	<input type="text"/>
Component Name:	<input type="text"/>	Command Description:	<input type="text"/>

Access Method

◆ **Find**

◆ Add

◆ Update

◆ Remove

Add / Update / Remove Password:

*Author: Dave Peters
dpeters@eos.hitc.com*

Figure 10.4.4.1-5. Command Description Specification Window

- E. **Command Prerequisite State Specification Record.** Open the Command Prerequisite State Specification window (see Figure 10.4.4.1-6) by clicking **Command Prerequisite State Specification** on the Command Parameters window.

Netscape: Command Prerequisite State Specification

File Edit View Go Bookmarks Options Directory Window Help

Back Forward Home Reload Images Open Print Find Stop

Location:

What's New What's Cool Handbook Net Search Net Directory Software

Flight Operations Segment

Command Prerequisite State Specification

◆ Command Prerequisite State

Command Mnemonic: Command Parameter Id:

Prerequisite Mnemonic: DN/EU Indicator:

Prerequisite Low Value: Prerequisite High Value:

Clear Form

Access Method

◆ Find

◆ Add

◆ Update

◆ Remove

Add / Update / Remove Password:

Submit

Author: Dave Peters
dpeters@eos.hitc.com

Figure 10.4.4.1-6. Command Prerequisite State Specification Window

The Command Prerequisite State Specification Record defines the condition for which a telemetry parameter associated with a command must occur in order to perform prerequisite state checking. Each command may specify up to 4 analog or discrete telemetry parameters with an associated range of values. A command will be transmitted only if all telemetry parameters defined for this command fall within their specified range. To indicate an exact value for the telemetry parameter, the low value and high value must be equal. The following is a list of fields associated with the Prerequisite State Specification:

1. **Command Parameter Identifier.** The number uniquely identifying the spacecraft or instrument command parameter. This value must be specified in the Command Parameter Specification PDB.
2. **Command Mnemonic.** The name of the command parameter. This value must also be specified in the Command Parameter Specification PDB in combination with the command parameter ID.
3. **Prerequisite Mnemonic.** The telemetry parameter whose value is used in determining whether the command is to be transmitted. This value must also be defined in the Telemetry Description PDB.
4. **DN/EU Indicator.** The units the prerequisite mnemonic is defined as (i.e., raw data number or engineering units), where:

DN = raw data number.

EU = engineering units.

This field is only used for analog telemetry parameters, therefore the telemetry parameter must be of type analog. Additionally, a prerequisite mnemonic expressed in engineering units must also have an associated definition in the Polynomial Coefficients Specification PDB.

5. **Prerequisite Low Value.** The lowest acceptable value, inclusive, of the prerequisite mnemonic to verify the condition for sending the associated command. This value cannot be greater than the high value. The format for this field is determined by the DN/EU indicator. A DN/EU indicator set to DN indicates this value will be defined as a raw data number and contains a decimal integer. A DN/EU indicator set to EU indicates this value will be defined in engineering units and contains a floating-point number.
 6. **Prerequisite High Value.** The highest acceptable value, inclusive, of the prerequisite mnemonic to verify the condition for sending the associated command. This value cannot be less than the CEV low value. The format for this field is determined by the DN/EU indicator. A DN/EU indicator set to DN indicates this value will be defined as a raw data number and contains a decimal integer. A DN/EU indicator set to EU indicates this value will be defined in engineering units and contains a floating-point number.
- F. **Command Variable Data Specification.** Open the Command Variable Data Specification window (see Figure 10.4.4.4.1-7) by clicking **Command Variable Data Specification** on the Command Parameters window.

The Command Variable Data Specification Record defines the subfields associated with variable type commands. Each subfield defines a parameter associated with a command to

be specified at execution time. Each command of variable type may reference up to 10 subfield names. Optional conversion equation and up to 10 state names may be associated with each subfield. A third order polynomial may be used to reverse calibrate an input subfield value from an EU to a DN using the following equation: $DN = C0 + C1*EU + C2*EU**2 + C3*EU**3$.

Netscape: Command Variable Data Word Specification

File Edit View Go Bookmarks Options Directory Window Help

Back Forward Home Reload Images Open Print Find Stop

Location:

What's New What's Cool Handbook Net Search Net Directory Software

Flight Operations Segment

Command Variable Data Specification

Command Variable Data

Command Mnemonic: Command Parameter Id:

Subfield Name: Default Value: Subfield Length: Min Subfield Value: Max Subfield Value:

Destination First Bit: Destination Last Bit: Units:

Conversion Group #1: Start EU #1: Conversion Group #2: Start EU #2:

State #1: State Value #1: State #2: State Value #2:

State #3: State Value #3: State #4: State Value #4:

State #5: State Value #5: State #6: State Value #6:

State #7: State Value #7: State #8: State Value #8:

State #9: State Value #9: State #10: State Value #10:

Clear Form

Access Method

☐ Find

☐ Add

☐ Update

Figure 10.4.4.4.1-7. Command Variable Data Specification Window

The following fields are included on the Command Variable Data Word Specification window:

1. **Command Parameter Identifier.** The number uniquely identifying the spacecraft or instrument command parameter. This value must be specified in the Command Parameter PDB.
2. **Command Mnemonic.** The name of the command parameter. This value must also be specified in the Command Parameter Specification PDB in combination with the command parameter ID.
3. **Subfield Name.** The name of a subfield associated with a variable type command.
4. **Default Value.** The value to be used if no value is specified when the command is issued. This value is represented as a Data Number (DN).
5. **Subfield Length.** The number of bits constituting the subfield value within the command bit pattern. Valid values are 1 to 32.
6. **Destination First Bit.** The first bit of the command bit pattern where the subfield value will be inserted in the command data message.
7. **Destination Last Bit.** The last bit of the command bit pattern where the subfield value will be inserted in the command data message.
8. **Minimum Value.** The lower bound for the associated state of the subfield value range.
9. **Maximum Value.** The upper bound of the subfield value range.
10. **Units.** The abbreviation for the engineering units to which a command subfield value is converted from.
11. **Conversion Group Number 1.** The number uniquely identifying a set of coefficients associated with an unsegmented 3rd order polynomial equation to be used when the subfield value is greater than or equal to the start EU number 1 (field 12), and less than the start EU number 2 (field 14). This value must be defined in the Polynomial Coefficients Specification PDB. The equation used to reverse calibrate is as follows:
$$DN = C0 + C1*EU + C2*EU**2 + C3*EU**3.$$
12. **Start EU number 1.** The lower bound of values for conversion group number 1.
13. **Conversion Group Number 2.** The number uniquely identifying a set of coefficients associated with a conversion equation to be used when the subfield value is greater than or equal to start EU number 2. This value must be defined in the Polynomial Coefficients Specification PDB.
14. **Start EU Number 2.** The lower bound of values for conversion group number 2.
15. **State Name 1.** The state associated with the subfield.
16. **State Value 1.** The value to be inserted into the command bit pattern when the user enters state name 1.
17. **State Names 2 to 10.** The state associated with subfields 2 to 10.
18. **State Values 2 to 10.** The value to be inserted into the command bit pattern when the user enters state names 2 to 10.